(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) Übersetzung der **europäischen Patentschrift**

(97) **EP 0 702 815 B1**

(10) **DE 694 25 658 T2**

(51) Int. Cl.[7]: **G 06 F 11/14**

(21) Deutsches Aktenzeichen: 694 25 658.7
(86) PCT-Aktenzeichen: PCT/US94/06320
(96) Europäisches Aktenzeichen: 94 921 242.7
(87) PCT-Veröffentlichungs-Nr.: WO 94/29807
(86) PCT-Anmeldetag: 2. 6. 1994
(87) Veröffentlichungstag der PCT-Anmeldung: 22. 12. 1994
(97) Erstveröffentlichung durch das EPA: 27. 3. 1996
(97) Veröffentlichungstag der Patenterteilung beim EPA: 23. 8. 2000
(47) Veröffentlichungstag im Patentblatt: 19. 4. 2001

(30) Unionspriorität:
71643 03. 06. 1993 US

(73) Patentinhaber:
Network Appliance, Inc., Sunnyvale, Calif., US

(74) Vertreter:
Klunker, Schmitt-Nilson, Hirsch, 80797 München

(84) Benannte Vertragstaaten:
AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

(72) Erfinder:
HITZ, David, Sunnyvale, US; MALCOM, Michael, Los Altos, US; LAU, James, Cupertino, US; RAKITZIS, Byron, Moutain View, US

(54) ANORDNUNG EINES DATEISYSTEMS ZUM BESCHREIBEN BELIEBIGER BEREICHE

DE 694 25 658 T2

HINTERGRUND DER ERFINDUNG

1. GEBIET DER ERFINDUNG

Die Erfindung betrifft das Gebiet von Verfahren und Vorrichtungen zum Unterhalten eines konsistenten Dateisystems und zum Schaffen von ausschließlich lesbaren Kopien des Dateisystems.

2. EINSCHLÄGIGER STAND DER TECHNIK

Sämtliche Dateisysteme müssen auch bei Systemausfall Konsistenz bewahren. Im Stand der Technik wurde zu diesem Zweck eine Reihe unterschiedlicher Konsistenzmethoden eingesetzt.

Eine der schwierigsten und zeitraubendsten Anforderungen bei der Verwaltung jedes Dateiservers ist die Anfertigung von Sicherungen der Dateidaten. Traditionelle Lösungen bestanden darin, eine Kopie der Daten auf Band oder andere Offline-Datenträger zu bringen. Bei einigen Dateisystemen muß beim Sicherungsprozeß der Datenserver offline gesetzt werden, um sicher zu gehen, daß der Sicherungsvorgang vollständig konsistent ist. Ein jüngerer Fortschritt bei der Datensicherung ist die Möglichkeit, ein Dateisystem rasch zu „klonen“ (d.i. ein zum Stand der Technik gehöriges Verfahren zum Erzeugen einer nur lesbaren Kopie des Dateisystems auf Platte), und eine Datensicherung anhand des Klons, und nicht aus dem aktiven Dateisystem zu erstellen. Bei diesem Typ von Datei kann der Datenserver beim Sicherungsbetrieb online bleiben.

Datenbank-Konsistenz

Eine herkömmliche Datenbank (Dateisystem) ist von Chutani, et al. offenbart in seinem Artikel mit dem Titel *The Episode File System*, USENIX, Winter 1992, Seiten 43-59. Dieser Artikel beschreibt das Episode-Dateisystem, bei dem es sich um eine Datenbank unter Verwendung von Meta-Daten (das heißt Inoden-Tabellen, Verzeichnissen, Momentaufnahmen und indirekten Blöcken) handelt. Es kann als eigenständige oder als verteilte Datenbank verwendet werden. *Episode* unterhält eine Mehrzahl separater Datenbank-Hirarchien. Episode nimmt kollektiv auf mehrere Datenbanken als „Aggregat" Bezug. Insbesondere schafft Episode einen Klon jeder Datenbank zur langsamen Änderung von Daten.

In Episode enthält jede logische Datenbank eine „Anoden"-Tabelle. Eine Anoden-Tabelle ist äquivalent einer in Datenbanken wie dem Berkeley Fast File System verwendeten Inoden-Tabelle. Es handelt sich um eine 252-Byte-Struktur. Anoden dienen zum Speichern sämtlicher Benutzerdaten sowie von Meta-Daten innerhalb des Episode-Dateisystems. Eine Anode beschreibt das Hauptverzeichnis einer Datenbank einschließlich Hilfsdateien und Verzeichnissen. Jedes derartige Dateisystem wird in Episode als eine „Dateimenge" (Fileset) referenziert. Sämtliche Daten innerhalb einer Dateimenge können geortet werden, indem iterativ durch die Anoden-Tabelle gegangen und jede Datei ihrerseits verarbeitet wird. Episode erzeugt eine ausschließlich lesbare Kopie einer Datenbank, die hier als „Klon" bezeichnet wird, und sie nutzt gemeinsam Daten mit dem aktiven Dateisystem unter Einsatz von Copy-On-Write-Methoden (COW-Methoden; Kopieren nach Schreiben).

Episode verwendet eine Protokollmethode zur Wiedererlangung einer oder mehrerer Datenbanken nach einem Systemzusammenbruch. Das Protokollieren garantiert, daß die Datei-Meta-Daten konsistent sind. Eine Momentaufnahmen-Tabelle enthält Information darüber, ob jeder Block innerhalb der Datenbank zugeordnet ist oder nicht. Außerdem zeigt die Momentaufnahmen-Tabelle an, ob jeder Block protokolliert ist oder nicht. Sämtliche

Meta-Daten-Aktualisierungen werden in einem Protokoll-„Behälter“ aufgezeichnet, der das Transaktions-Protokoll des Aggregats speichert. Das Protokoll wird als Kreispuffer von Platten-Blöcken verarbeitet. Die Transaktions-Protokollierung von Episode verwendet Protokolliermethoden, die ursprünglich für Datenbanken mit dem Zweck entwickelt wurden, Dateisystem-Konsistenz zu garantieren. Diese Methode macht sorgfältigen Gebrauch von Schreibbefehlen sowie einem Wiederherstellungsprogramm, die von Datenbankmethoden innerhalb des Wiederherstellungsprogramms unterstützt werden.

Andere zum Stand der Technik zählende Systeme enthalten JFS von IBM und VxFS von Veritas Corporation und machen Gebrauch von unterschiedlichen Formen der Transaktions-Protokollierung, um den Wiederherstellungsprozeß zu beschleunigen, allerdings erfordern sie immer noch einen Wiederherstellungsprozeß.

Ein weiteres bekanntes Verfahren wird als Methode des „geordneten Schreibens“ bezeichnet. Es schreibt sämtliche Platten-Blöcke in sorgfältig festgelegter Reihenfolge, so daß Schaden minimiert wird, wenn es zu einem Systemausfall kommt, während eine Reihe von zueinander in Beziehung stehender Schreibvorgänge durchgeführt wird. Dieser Stand der Technik versucht sicherzustellen, daß möglicherweise auftretende Inkonsistenzen harmlos sind. Beispielsweise werden einige wenige ungenutzte Blöcke oder Inoden als zugeordnet markiert. Der Hauptnachteil dieser Methode besteht darin, daß die dadurch der Plattenordnung auferlegten Restriktionen eine hohe Leistungsfähigkeit kaum zulassen.

Ein weiteres bekanntes System ist eine Weiterentwicklung des zweiten bekannten Verfahrens, bezeichnet als Methode des „geordneten Schreibens mit Wiederherstellung“. Bei diesem Verfahren können Inkonsistenzen möglicherweise schädlich sein. Allerdings ist die Reihenfolge von Schreibvorgängen derart beschränkt, daß sich Inkonsistenzen auffinden und mit Hilfe eines Wiederherstellungsprogramms fixieren lassen. Beispiele für dieses Verfahren umfassen das ursprüngliche UNIX-Dateisystem

sowie das Berkeley Fast File System (FFS). Diese Methode verringert die Platten-Reihenfolge ausreichend, um die Leistungseinbuße der Platten-Auftragserteilung zu beseitigen. Ein weiterer Nachteil besteht darin, daß der Wiederherstellungsprozeß zeitraubend ist. Typischerweise ist er proportional zur Größe des Dateisystems. Die Wiederherstellung eines 5 GB umfassenden FFS-Dateisystems erfordert daher beispielsweise zur Durchführung eine Stunde oder mehr.

Dateisystem-Klone

Figur 1 ist ein den Stand der Technik zeigendes Diagramm für das Episode-Dateisystem und veranschaulicht den Einsatz von Kopieren-Nach-Schreiben-Methoden (COW-Methoden), um einen Dateimengen-Klon zu erzeugen. Eine Anode 110 enthält einen ersten Zeiger (Pointer) 110A mit einem gesetzten COW-Bit. Der Zeiger 110A referenziert den Datenblock 114 direkt. Die Anode 110 enthält einen zweiten Zeiger 110B, dessen COW-Bit gelöscht ist. Der Zeiger 110B der Anode referenziert den Block 112 indirekt. Der indirekte Block 112 enthält einen Zeiger 112A, der den Datenblock 124 direkt referenziert. Das COW-Bit des Zeigers 112A ist gesetzt. Der indirekte Block 112 enthält einen zweiten Zeiger 112B, der den Datenblock 126 referenziert. Das COW-Bit des Zeigers 112B ist gelöscht.

Eine Klon-Anode 120 enthält einen ersten Zeiger 120A, der auf den Datenblock 114 zeigt. Das COW-Bit des Zeigers 120A ist gelöscht. Der zweite Zeiger 120B der Klon-Anode 120 referenziert den indirekten Block 122. Das COW-Bit des Zeigers 120B ist gelöscht. Der indirekte Block 122 enthält einen Zeiger 122A, der den Datenblock 124 referenziert. Das COW-Bit des Zeigers 122A ist gelöscht.

Wie in Figur 1 gezeigt ist, enthält jeder direkte Zeiger 110A, 112A-112B, 120A und 122A und jeder indirekte Zeiger 110B und 120B in dem Episode-Dateisystem ein COW-Bit. Blöcke, die nicht modifiziert wurden, sind sowohl im aktiven Dateisystem als auch in dem Klon enthalten, und bei

ihnen sind die COW-Bits gesetzt (1). Das COW-Bit ist gelöscht (0), wenn ein von dem Zeiger referenzierter Block modifiziert wurde und damit Teil des aktiven Dateisystems ist, nicht jedoch Teil des Klons.

Wird ein Kopieren-Nach-Schreiben-Block modifiziert, so wird gemäß Figur 1 ein neuer Block zugeordnet und aktualisiert. Das COW-Flag in dem Zeiger auf diesen neuen Block wird dann gesetzt. Das COW-Bit des Zeigers 110A der ursprünglichen Anode 110 wird gelöscht. Wenn also die Klon-Anode 120 erzeugt wird, referenziert die Klon-Anode 120 auch den Datenblock 114. Sowohl die Original-Anode 110 als auch die Klon-Anode 120 referenzieren den Datenblock 114. Auch der Datenblock 124 wurde modifiziert, angedeutet durch ein gelöschtes COW-Bit des Zeigers 112A in dem ursprünglichen indirekten Block 112. Wenn folglich die Klon-Anode erzeugt wird, wird der indirekte Block 122 erzeugt. Der Zeiger 122A des indirekten Blocks 122 referenziert den Datenblock 124, und das COW-Bit des Zeigers 122A ist gelöscht. Sowohl der indirekte Block 122 der Original-Anode 110 als auch der indirekte Block 122 der Klon-Anode 120 referenzieren den Datenblock 124.

Figur 1 zeigt das Kopieren einer Anode zum Erzeugen einer Klon-Anode 120 für eine einzelne Datei. Allerdings müssen Klon-Anoden für jede Datei erzeugt werden, die geänderte Datenblöcke in dem Dateisystem enthält. Zur Zeit des Klonens müssen sämtliche Inoden kopiert werden. Das Erzeugen von Klon-Anoden für jede modifizierte Datei innerhalb des Dateisystems kann signifikante Mengen an Plattenspeicherplatz verbrauchen. Außerdem ist Episode nicht in der Lage, Mehrfach-Klone zu handhaben, da jeder Zeiger lediglich ein einziges COW-Bit aufweist. Ein einzelnes COW-Bit ist nicht in der Lage, mehr als einen Klon zu unterscheiden. Bei mehr als einem Klon gibt es kein zweites COW-Bit, welches gesetzt werden könnte.

Ein Dateisatz „Klon“ ist eine ausschließlich lesbare Kopie eines aktiven Dateisatzes, wohingegen der aktive Dateisatz selbst sowohl lesbar als auch beschreibbar ist. Klone werden unter Verwendung von COW-Methoden

implementiert und nutzen gemeinsam Datenblöcke mit einem aktiven Dateisatz auf Block-Für-Block-Basis. Episode implementiert das Klonen dadurch, daß jede in einem Dateisatz gespeicherte Anode kopiert wird. Nach dem anfänglichen Klonen zeigen sowohl die beschreibbare Anode des aktiven Dateisatzes als auch die geklonte Anode auf denselben Datenblock oder dieselben Datenblöcke. Allerdings sind Plattenadressen für direkte und indirekte Blöcke innerhalb der Original-Anode als COW gekennzeichnet. Deshalb hat eine Aktualisierung des beschreibbaren Dateisatzes keinen Einfluß auf den Klon. Wird ein COW-Block modifiziert, so wird ein neuer Block in dem Dateisystem zugewiesen und mit der Modifizierung aktualisiert. Das COW-Flag in dem Zeiger dieses neuen Blocks wird gelöscht. Das bekannte Episode-System erzeugt Klone, die die gesamte Inoden-Datei sowie sämtliche indirekten Blöcke innerhalb des Dateisystems duplizieren. Episode dupliziert sämtliche Inoden und indirekten Blöcke derart, daß es ein Kopieren-Nach-Schreiben-(COW-)Bit in sämtlichen Zeigern auf Blöcke setzen kann, die sowohl von dem aktiven Dateisystem als auch von dem Klon benutzt werden. Bei Episode ist es wichtig, diese Blöcke zu kennzeichnen, so daß neue, in das aktive Dateisystem eingeschriebene Daten die alten Daten, welche Teil des Klons sind, und die deshalb nicht geändert werden dürfen, nicht überschreiben.

Das Erzeugen eines Klons im Stand der Technik kann bis zu 32 MB auf eine 1-GB-Platte verbrauchen. Der Stand der Technik verwendet 256 MB Plattenspeicherraum auf einer 1-GB-Platte (für 4-KB-Blöcke), um acht Klone des Dateisystems zu halten. Damit kann der Stand der Technik keine großen Anzahlen von Klonen zum Verhindern von Datenverlusten verwenden. Statt dessen erleichterte er üblicherweise das Sichern des Dateisystems auf eine Hilfsspeichereinrichtung, verschieden von dem Plattenlaufwerk, so zum Beispiel ein Band-Sicherungsgerät. Klone werden zum Sichern eines Dateisystems in einem konsistenten Zustand in dem Zeitpunkt verwendet, zu dem der Klon hergestellt wird. Durch Klonen des Dateisystems kann der Klon zur Sicherheit auf das Hilfsspeichersystem gebracht werden, ohne daß dabei das aktive Dateisystem abgeschaltet wird, wodurch Benutzer an einer Benutzung des Dateisystems gehindert würden.

Damit ermöglichen es Klone den Benutzern, weiterhin auf ein aktives Dateisystem zuzugreifen, während das Dateisystem selbst in einem konsistenten Zustand gesichert wird. Anschließend wird der Klon gelöscht, nachdem die Sicherung abgeschlossen ist. Episode ist nicht in der Lage, mehrere Klone zu führen, da jeder Zeiger nur ein COW-Bit enthält. Ein einzelnes COW-Bit ist nicht im Stande, mehr als einen Klon zu unterscheiden. Bei mehr als einem Klon gibt es kein zweites COW-Bit, welches gesetzt werden könnte.

Ein Nachteil des bekannten Systems zum Erzeugen von Dateisystem-Klonen besteht darin, daß das System sämtliche Inoden und sämtliche indirekten Blöcke innerhalb des Dateisystems dupliziert. Bei einem System mit zahlreichen kleinen Dateien können die Inoden allein einen signifikanten Prozentsatz des gesamten Plattenspeicherraums eines Dateisystems belegen. In einem 1-GB-Dateisystem zum Beispiel, welches mit 4-KB-Dateien gefüllt ist, gibt es 32 MB Inoden. Das Erzeugen eines Episode-Klons verbraucht also einen signifikanten Anteil des Plattenspeicherraums und erzeugt große Mengen (das heißt zahlreiche Megabytes) an Plattenverkehr. Als Ergebnis dieser Zustände nimmt das Erzeugen eines Klons eines Dateisystems einen beträchtlichen Zeitraum bis zur Vervollständigung in Anspruch.

Ein weiterer Nachteil des bekannten Systems besteht darin, daß das System die Erzeugung mehrerer Klone desselben Dateisystems schwierig macht. Im Ergebnis neigen die Klone dazu, einzeln für Kurzzeitoperationen verwendet zu werden, so zum Beispiel zum Sichern des Dateisystems auf Band, um dann gelöscht zu werden.

Die in den geänderten Ansprüchen definierte Erfindung schafft ein Verfahren zum Halten eines Dateisystems in einem konsistenten Zustand sowie zum Erzeugen von ausschließlich lesbaren Kopien eines Dateisystems. Änderungen des Dateisystems werden streng gesteuert, um das Dateisystem in einem konsistenten Zustand zu halten. Das Dateisystem schreitet von einem selbst-konsistenten Zustand zu einem weiteren selbst-

konsistenten Zustand weiter. Die Menge an selbst-konsistenten Blöcken auf einer Platte, die durch die Haupt-Inode beherrscht wird, wird als Konsistenzpunkt (CP) bezeichnet. Zum Implementieren von Konsistenzpunkten schreibt WAFL stets neue Daten in nicht-zugewiesene Blöcke auf der Platte. Es überschreibt niemals existierende Daten. Ein neuer Konsistenzpunkt tritt auf, wenn der Finsfo-Block dadurch aktualisiert wird, daß eine neue Haupt-Inode für die Inodendatei in ihn eingeschrieben wird. Solange die Haupt-Inode nicht aktualisiert wird, ändert sich also der Zustand des Dateisystems auf der Platte nicht.

Die vorliegende Erfindung schafft außerdem Schnappschüsse, bei denen es sich um virtuelle, ausschließlich lesbare Kopien des Dateisystems handelt. Ein Schnappschuß nimmt keinen Plattenspeicherplatz in Anspruch, wenn er am Anfang erzeugt wird. Er ist derart ausgestaltet, daß zahlreiche verschiedene Schnappschüsse für ein und dasselbe Dateisystem erzeugt werden können. Im Gegensatz zu herkömmlichen Dateisystemen, die einen Klon durch Duplizieren des gesamten Inoden-Dateisatzes und sämtlicher indirekter Blöcke duplizieren, dupliziert die vorliegende Erfindung nur diejenige Inode, die die Inodendatei beschreibt. Der also tatsächlich benötigte Plattenspeicherplatz für eine Momentaufnahme beträgt lediglich 128 Bytes, die zum Speichern der duplizierten Inode verwendet werden. Die 128 Bytes, die erfindungsgemäß für eine Momentaufnahme oder einen Schnappschuß benötigt werden, sind deutlich weniger als die zahlreichen Megabytes, die für einen Klon im Stand der Technik benötigt werden.

Die vorliegende Erfindung verhindert, daß neue Daten, die in das aktive Dateisystem geschrieben werden, „alte“ Daten, die Teil eines oder mehrerer Schnappschüsse sind, überschreiben. Notwendig ist, daß alte Daten solange nicht überschrieben werden, wie sie Teil eines Schnappschusses sind. Erreicht wird dies durch Verwendung einer freien Mehrfachbit-Blockabbildung. Die meisten zum Stand der Technik gehörigen Dateisysteme verwenden eine freie Blockabbildung mit einem einzelnen Bit pro Block, um anzugeben, ob ein Block zugewiesen ist oder nicht. Die vorliegende Erfindung verwendet eine Blockabbildung mit 32-Bit-Einträgen. Ein

erstes Bit gibt an, ob ein Block von dem aktiven Dateisystem verwendet wird, und 20 verbleibende Bits werden für bis zu 20 Schnappschüsse verwendet, allerdings können einige Bits der 31 Bits für andere Zwecke verwendet werden.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

Figur 1 ist ein Blockdiagramm eines zum Stand der Technik zählenden „Klons" eines Dateisystems.

Figur 2 ist ein Diagramm, welches eine Liste von Inoden mit unsauberen Puffern veranschaulicht.

Figur 3 ist ein Diagramm, das eine platteninterne Inode des WAFL darstellt.

Figuren 4A-4D sind Diagramme, die platteninterne Inoden von WAFL mit unterschiedlichen Umwege-Ebenen veranschaulichen.

Figur 5 ist ein Flußdiagramm des Verfahrens zum Erzeugen eines Konsistenzpunkts.

Figur 6 ist ein Flußdiagramm zur Veranschaulichung des Schritts 530 aus Figur 5 zum Erzeugen eines Konsistenzpunkts.

Figur 7 ist ein Flußdiagramm zum Veranschaulichen des Schritts 530 in Figur 5 zum Erzeugen eines Schnappschusses.

Figur 8 ist ein Diagramm zum Veranschaulichen einer Intern-Inode des WAFL gemäß der Erfindung.

Figur 9A-9D sind Diagramme, die Intern-Inoden des WAFL mit unterschiedlichen Umwege-Ebenen gemäß der Erfindung darstellen.

Figur 10 ist ein Diagramm zum Veranschaulichen einer Intern-Inode 1020 für eine Datei.

Figuren 11A-11D sind Diagramme zum Veranschaulichen einer Blockabbilddatei (blkmap) gemäß der Erfindung.

Figur 12 ist ein Diagramm zum Veranschaulichen einer erfindungsgemäßen Inoden-Datei.

Figuren 13A-13B sind Diagramme zum Veranschaulichen einer Inodenabbild-Datei (inomap) gemäß der Erfindung.

Figur 14 ist ein Diagramm zum Veranschaulichen eines erfindungsgemäßen Verzeichnisses.

Figur 15 ist ein Diagramm zum Veranschaulichen einer Dateisysteminformationsstruktur (fsinfo).

Figur 16 ist ein Diagramm zum Veranschaulichen des WAFL-Dateisystems.

Figuren 17A-17L sind Diagramme zum Veranschaulichen des Erzeugens eines Konsistenzpunkts.

Figuren 18A-18C sind Diagramme zum Veranschaulichen des Erzeugens eines Schnappschusses.

Figur 19 ist ein Diagramm zum Veranschaulichen von Änderungen einer Inodendatei.

Figur 20 ist ein Diagramm zum Veranschaulichen von fsinfo-Blöcken, die zum Halten eines Dateisystems in einem konsistenten Zustand verwendet werden.

Figuren 21A-21F sind detaillierte Diagramme zum Veranschaulichen des Erzeugens eines Schnappschusses.

Figur 22 ist ein Diagramm zum Veranschaulichen eines aktiven WAFL-Dateisystems mit drei Schnappschüssen, die jeweils eine gemeinsame Datei referenzieren; und

Figuren 23A-23B sind Diagramme zum Veranschaulichen der Aktualisierung einer Zugriffszeit.

DETAILLIERTE BESCHREIBUNG DER ERFINDUNG

Beschrieben wird ein System zum Erzeugen von ausschließlich lesbaren Kopien eines Dateisystems (einer Datenbank). In der folgenden Beschreibung werden zahlreiche spezifische Einzelheiten, so zum Beispiel Anzahl und Beschaffenheit von Platten, Plattenblock-Größen etc. im einzelnen beschrieben, um eine ausführlichere Beschreibung der Erfindung anzubieten. Es ist jedoch für den Fachmann ersichtlich, daß die Erfindung auch ohne diese spezifischen Einzelheiten ausgeführt werden kann. Andererseits wurden bekannte Merkmale nicht im einzelnen beschrieben, um die Erfindung nicht in unnötiger Weise zu verundeutlichen.

WRITE-ANYWHERE-DATEISYSTEM-LAYOUT

Die vorliegende Erfindung macht Gebrauch von einem Write-Anywhere-Dateisystem-Layout (WAFL von Write Anywhere File-system Layout), also von einer Dateisystem-Konfiguration, die ein Aufzeichnen oder Schreiben an beliebiger Stelle ermöglicht. Das Plattenformatsystem beruht auf Blöcken (das heißt 4 KB Blöcken, die keine Fragmente besitzen), verwendet Inoden zum Beschreiben seiner Dateien, und enthält Verzeichnisse, die einfach speziell formatierte Dateien sind. WAFL verwendet Dateien zum Speichern von Meta-Daten, welche das Layout des Dateisystems beschreiben. Die WAFL-Meta-Dateien beinhalten: eine Inodendatei, eine Blockabbild-Datei (blkmap) und eine Inodenabbilddatei (inomap). Die

Inodendatei enthält die Inodentabelle für das Dateisystem. Die blkmap-Datei gibt an, welche Platten-Blöcke zugeordnet sind. Die inomap-Datei gibt an, welche Inoden zugeordnet sind. Weiter unten werden Unterscheidungsmerkmale für plattenintemе und WAFL-interne Inoden diskutiert.

Plattenintemе WAFL-Inoden

WAFL-Inoden unterscheiden sich von herkömmlichen Inoden. Jede WAFL-Inode verweist auf 16 Blöcke mit gleicher Umwegeebene. Eine Blocknummer ist 4 Bytes lang. Die Verwendung von Blocknummern mit gleicher Umwegeebene in einer Diode erleichtert die rekursive Verarbeitung einer Datei. Figur 3 ist ein Blockdiagramm, das eine plattenintemе Inode 310 veranschaulicht. Die plattenintemе Inode 310 besteht aus Standard-Inodeninformation 310A sowie 16 Blocknummern-Einträgen 310B gleicher Umwegeebene. Die Inodeninformation 310A umfaßt Information über den Inhaber einer Datei, Berechtigungen, Dateigröße, Zugriffszeit, etc., wie dies dem Fachmann alles bekannt ist. Im Gegensatz zu bekannten Inoden, die eine Mehrzahl von Blocknummern unterschiedlicher Umwegeebenen aufweisen, ist die plattenintemе Inode 310 anders. Durch Halten sämtlicher Blocknummerneinträge 310B innerhalb einer Inode 310 auf gleicher Umwegeebene wird die Implementierung des Dateisystems vereinfacht.

Für eine kleine Datei mit einer Größe von 64 Bytes oder weniger werden Daten direkt in der Inode selbst anstatt in Form von 16 Blocknummern gespeichert. Figur 4A ist ein Diagramm, das eine Inode 410 der Ebene 0 veranschaulicht, die der in Figur 3 gezeigten Inode 310 ähnelt. Allerdings enthält die Inode 410 64 Bytes Daten 410B anstelle von 16 Blocknummern 310B. Deshalb brauchen Plattenblöcke bei sehr kleinen Dateien nicht zugewiesen zu werden.

Für eine Datei mit einer Größe von weniger als 64 KB nimmt jede der 16 Blocknummern direkt auf einen 4-KB-Datenblock Bezug. Figur 4B ist ein Diagramm, welches eine Inode 310 der Ebene 1 mit 16 Blocknummern

310B veranschaulicht. Die Blocknummerneinträge 0-15 verweisen auf entsprechende 4-KB-Datenblöcke 420A-420C.

Für eine Datei mit einer Größe, die gleich oder größer ist als 64 KB und kleiner als 64 MB ist, nimmt jede der 16 Blocknummern Bezug auf einen einfach-indirekten Block. Seinerseits enthält jeder einzeln indirekte 4-KB-Block 1024 Blocknummern, die 4 KB-Datenblöcke referenzieren. Figur 4C ist ein Diagramm, welches eine Inode der Ebene 4, 310, veranschaulicht, welche 16 Blocknummern 310B enthält, die 16 einfach-indirekte Blöcke 430A-430C referenzieren. Wie in Figur 4C gezeigt ist, zeigt der Blocknummerneintrag 0 auf einen einfach-indirekten Block 430A. Der einfach-indirekte Block 430A enthält 1024 Blocknummern, die auf 4-KB-Datenblöcke 440A-440C Bezug nehmen. In ähnlicher Weise kann jeder einfach-indirekte Block 430B-430C jeweils bis zu 1024 Datenblöcke adressieren.

Bei einer Dateigröße von mehr als 64 MB referenzieren die 16 Blocknummern der Inode doppelt-indirekte Blöcke. Jeder doppelt-indirekte 4-KB-Block enthält 1024 Blocknummern, die auf entsprechende einfach-indirekte Blöcke verweisen. Jeder einfach-indirekte Block wiederum enthält 1024 Blocknummern, die auf 4-KB-Datenblöcke zeigen. Auf diese Weise lassen sich bis zu 64 GB adressieren. Figur 4D ist ein Diagramm einer Inode 310 der Ebene 3, die 16 Blocknummern 310B enthält, wobei Blocknummerneinträge 0, 1 und 15 auf doppelt-indirekte Blöcke 470A, 470B und 470C verweisen. Der doppelt-indirekte Block 470A enthält 1024 Blocknummerneinträge 0-1023, die auf 1024 einfach-indirekte Blöcke 480A-480B zeigen. Jeder einfach-indirekte Block 480A-480B wiederum referenziert 1024 Datenblöcke. Wie in Figur 4D gezeigt ist, referenziert der einfach-indirekte Block 480A 1024 Datenblöcke 490A-490C, und der einfach-indirekte Block 480B nimmt Bezug auf 1024 Datenblöcke 490C-490F.

WAFL-interne Inoden

Figur 8 ist ein Blockdiagramm, das eine WAFL-interne Inode 820 veranschaulicht. Die interne Inode 820 enthält die Information der plattenineternen Inode 310 (dargestellt in Figur 3), eine WAFL-Puffer-Datenstruktur 820A, außerdem 16 Pufferzeiger 820B. Eine WAFL-interne Inode besitzt eine Größe von 300 Bytes. Ein WAFL-Puffer ist ein 4 KB umfassendes (speicher-)internes Äquivalent der 4-KB-Blöcke, die auf der Platte gespeichert sind. Die Intern-Inode 820 unterscheidet sich von herkömmlichen Inoden, welche Puffer mit unterschiedlichen Umwegeebenen referenzieren. Jede Intern-WAFL-Inode 820 zeigt auf 16 Puffer mit gleicher Umwegeebene. Ein Pufferzeiger hat eine Länge von 4 Bytes. Indem man sämtliche Pufferzeiger 820B in eine Inode 820 auf der gleichen Umwegeebene hält, vereinfacht man die Dateisystem-Implementierung. Die Intern-Inode 820 enthält außerdem Intern-Information 820C, umfassend ein „Unsauber"-Flag, ein Inkonsistenzpunkt-Flag (IN_CP) sowie Zeiger für eine Verknüpfungsliste. Das Unsauber-Flag gibt an, daß die Inode selbst modifiziert wurde oder daß sie Puffer referenziert, welche ihrerseits geändert wurden. Das IN_CP-Flag dient zum Markieren einer Inode als in einem Konsistenzpunkt befindlich (wird unten beschrieben). Die Zeiger für eine verknüpfte Liste werden unten beschrieben.

Figur 10 ist ein Diagramm, welches eine Datei veranschaulicht, die durch eine WAFL-Inode 1010 referenziert wird. Die Datei enthält indirekte WAFL-Puffer 1020-1024 und direkte WAFL-Puffer 1030-1034: die WAFL-Intern-Inode 1010 enthält Standard-Inoden-Information 1010A (einschließlich eines Zählers für unsaubere Puffer), eine WAFL-Pufferdatenstruktur 1010B, 16 Pufferzeiger 1010C und eine standardmäßige plattenineterne Inode 1010D. Die interne WAFL-Inode 1010 hat eine Größe von etwa 300 Bytes. Die plattenineterne Inode hat eine Größe von 128 Bytes. Die WAFL-Pufferdatenstruktur 1010B umfaßt zwei Zeiger, von denen der erste die 16 Pufferzeiger 1010C und der zweite plattenineterne Blocknummern 1010D referenziert.

Jede Inode 1010 besitzt eine Zählung von unsauberen Puffern, auf die sie Bezug nimmt. Eine Inode 1010 kann in die Liste unsauberer Inoden und/oder die Liste von Inoden eingegeben werden, welche unsaubere Puffer aufweisen. Wenn sämtliche von einer Inode referenzierten unsauberen Puffer für die Aufzeichnung auf Platte vorgesehen sind oder auf Platte aufgezeichnet werden, wird die Zählung der unsauberen Puffer für Inode 1010 auf Null gesetzt. Die Inode 1010 wird dann entsprechend ihrem Flag neu in Warteschlange gestellt (das heißt in diesem Fall gibt es keine unsauberen Puffer). Diese Inode 1010 wird gelöscht, bevor die nächste Inode verarbeitet wird. Außerdem wird das Flag der Inode gelöscht, welches angibt, daß die Inode sich in einem Konsistenzpunkt befindet. Die Inode 1010 selbst wird in einem Konsistenzpunkt auf Platte geschrieben.

Die WAFL-Pufferstruktur ist dargestellt durch einen indirekten WAFL-Puffer 1020. Der WAFL-Puffer 1020 enthält eine WAFL-Pufferdatenstruktur 1020A, einen 4-KB-Puffer 1020B mit 1024 WAFL-Pufferzeigern und einen 4-KB-Puffer 1020C mit 1024 plattenintemen Blocknummern. Die WAFL-Pufferdatenstruktur hat eine Größe von 56 Bytes und enthält zwei Zeiger. Ein Zeiger der WAFL-Pufferdatenstruktur 1020A referenziert den 4-KB-Puffer 1020B, und ein zweiter Zeiger referenziert den Puffer 1020C. In Figur 10 zeigen die 16 Pufferzeiger 1010C der WAFL-Inode 1010 auf die 16 einfach-indirekten WAFL-Puffer 1020-1024. Der WAFL-Puffer 1020 wiederum referenziert 1024 direkte WAFL-Pufferstrukturen 1030-1034. Der WAFL-Puffer 1030 steht repräsentativ für direkte WAFL-Puffer.

Der direkte WAFL-Puffer 1030 enthält eine WAFL-Pufferdatenstruktur 1030A und einen 4-KB-Direktpuffer 1030B, der eine gecachete Version eines entsprechenden plattenintemen 4-KB-Datenblocks enthält. Der direkte WAFL-Puffer 1030 enthält nicht einen 4-KB-Puffer wie den Puffer 1020C des Indirekt-WAFL-Puffers 1020. Der zweite Pufferzeiger der WAFL-Pufferdatenstruktur 1030A wird auf Null gesetzt und zeigt daher nicht auf einen zweiten 4-KB-Puffer. Dies verhindert eine nicht effiziente

Verwendung von Speicher, da ansonsten Speicherbereich für einen unbenutzten Puffer bereitgestellt würde.

In einem WAFL-Dateisystem, wie es in Figur 10 gezeigt ist, referenziert eine interne WAFL-Inodenstruktur 1010 einen Baum von WAFL-Pufferstrukturen 1020-1024 und 1030-1034. Dieser ähnelt einem Baum von plattenintemen Blöcken, die durch Standard-Inoden referenziert werden, welche Blocknummern aufweisen, die auf indirekte und/oder direkte Blöcke zeigen. Damit enthält die WAFL-Inode 1010 nicht nur die 16 Volumen-Blocknummern enthaltende platteninterne Inode 1010D, sondern enthält außerdem 16 Pufferzeiger 1010C, welche auf WAFL-Pufferstrukturen 1020-1024 und 1030-1034 zeigen. WAFL-Puffer 1030-1034 enthalten gecachete Inhalte von Blöcken, die durch Volumen-Blocknummern referenziert werden.

Die WAFL-Intern-Inode 1010 enthält 16 Pufferzeiger 1010C. Ihrerseits werden die 16 Pufferzeiger 1010C durch eine WAFL-Pufferstruktur 1010B referenziert, die die Wurzel für den Baum aus WAFL-Puffern 1020-1024 und 1030-1034 bildet. Somit enthält jede WAFL-Inode 1010 eine WAFL-Pufferstruktur 1010B, die auf die 16 Pufferzeiger 1010C innerhalb der Inode 1010 zeigt. Dies erleichtert die rekursive Implementierung von Algorithmen zum Handhaben von Puffer-Bäumen. Wenn die 16 Pufferzeiger 1010C innerhalb der Inode 1010 nicht durch eine WAFL-Pufferstruktur 1010B repräsentiert würden, ließe sich der rekursive Algorithmus zum Bearbeiten des gesamten Baums von Puffern 1020-1024 und 1030-1034 nur schwer implementieren.

Figuren 9A-9D sind Diagramme, die Inoden mit unterschiedlichen Umwegeebenen zeigen. In Figuren 9A-9D sind zur Darstellung der Indirektheit oder der Umwege indirekte und direkte WAFL-Puffer dargestellt. Allerdings sollte gesehen werden, daß die WAFL-Puffer in Figur 9 entsprechende indirekte oder direkte Puffer aus Figur 10 repräsentieren. Bei einer kleinen Datei mit einer Größe von 64 Bytes oder weniger werden Daten direkt in der Inode selbst gespeichert, und nicht die 16 Pufferzeiger. Figur

9A ist ein Diagramm, das eine Inode 820 der Ebene Null veranschaulicht, bei der es sich um die gleiche Inode handelt wie die Inode 820 in Figur 8, nur daß die Inode 820 an Stelle von 16 Pufferzeigern 820B nunmehr 64 Datenbytes 920B enthält. Deshalb werden bei sehr kleinen Dateien keine zusätzlichen Puffer zugewiesen.

Bei einer Datei mit einer Größe von weniger als 64 KB referenziert jeder der 16 Pufferzeiger direkt einen direkten 4-KB-WAFL-Puffer. Figur 9B ist ein Diagramm einer Inode 820 der Ebene 1 mit 16 Pufferzeigern 820B. Die Pufferzeiger PTR0-PTR15 zeigen auf entsprechende direkte 4-KB-WAFL-Puffer 922A-922C.

Bei einer Datei, die größer oder gleich 64 KB und kleiner als 64 MB ist, referenziert jeder der 16 Pufferzeiger einen einfach-indirekten WAFL-Puffer. Jeder einfach-indirekte 4-KB-WAFL-Puffer seinerseits umfaßt 1024 Pufferzeiger, welche 4-KB-Direkt-WAFL-Puffer referenzieren. Figur 9C ist ein Diagramm einer Inode 820 der Ebene 2 mit 16 Pufferzeigern 820B, welche 16 einfach-indirekte WAFL-Puffer 930A-930C referenzieren. Gemäß Figur 9C zeigt der Pufferzeiger PTR0 auf einen einfach-indirekten WAFL-Puffer 930A. Der einfach-indirekte WAFL-Puffer 930A enthält 1024 Zeiger, die 4-KB-Direkt-WAFL-Puffer 940A-940C referenzieren. In ähnlicher Weise können einfach-indirekte WAFL-Puffer 930B-930C jeweils bis zu 1024 direkte WAFL-Puffer adressieren.

Bei einer Dateigröße von mehr als 64 MB referenzieren die 16 Pufferzeiger der Inode doppelt-indirekte WAFL-Puffer. Jeder 4 KB umfassende, doppelt-indirekte WAFL-Puffer enthält 1024 Zeiger, die auf zugehörige einfach-indirekte WAFL-Puffer zeigen. Jeder einfach-indirekte WAFL-Puffer seinerseits umfaßt 1024 Zeiger, die auf direkte 4 KB-WAFL-Puffer zeigen. Damit können bis zu 64 GB adressiert werden. Figur 9D ist ein Diagramm einer Inode 820 der Ebene 3 mit 16 Zeigern 820B, wobei Zeiger PTR0, PTR1 und PTR15 doppelt-indirekte WAFL-Puffer 970A, 970B bzw. 970C referenzieren. Der doppelt-indirekte WAFL-Puffer 970A enthält 1024 Zeiger, die auf 1024 einfach-indirekte WAFL-Puffer 980A-980B

zeigen. Jeder einfach-indirekte WAFL-Puffer 980A-980B wiederum referenziert 1024 direkte WAFL-Puffer. Wie in Figur 9D zu sehen ist, referenziert der einfach-indirekte WAFL-Puffer 980A 1024 direkte WAFL-Puffer 990A-990C, und der einfach-indirekte WAFL-Puffer 980B referenziert 1024 direkte WAFL-Puffer 990D-990F.

Verzeichnisse

Verzeichnisse innerhalb des WAFL-Systems sind in 4-KB-Blöcken gespeichert, welche in zwei Abschnitte aufgeteilt sind. Figur 14 ist ein Diagramm, das einen Verzeichnisblock 1410 gemäß der Erfindung veranschaulicht. Jeder Verzeichnisblock 1410 enthält einen ersten Abschnitt 1410A mit Verzeichniseintrag-Strukturen 1412-1414 fester Länge, und einen zweiten Abschnitt 1410B, der die aktuellen Verzeichnisnamen 1416-1418 enthält. Jeder Verzeichniseintrag enthält außerdem eine Datei-ID, das heißt eine Datei-Kennung und eine Generation. Diese Information kennzeichnet, welche Datei der Eintrag referenziert. Diese Information ist im Stand der Technik bekannt und deshalb in Figur 14 nicht dargestellt. Jeder Eintrag 1412-1414 im ersten Abschnitt 1410A des Verzeichnisblocks besitzt einen Zeiger auf seinen Namen innerhalb des zweiten Abschnitts 1410B. Außerdem enthält jeder Eintrag 1412-1414 einen Hash-Wert, abhängig von seinem Namen in dem zweiten Abschnitt 1410B, so daß der Name nur untersucht wird, wenn es zu einem Hash-Treffer (einer Hash-Übereinstimmung) kommt. Beispielsweise enthält der Eintrag 1412 des ersten Abschnitts 1410A einen Hash-Wert 1412A und einen Zeiger 1412B. Der Hash-Wert 1412A ist ein Wert, der von dem Verzeichnis-Namen „VERZEICHNIS_ABC“ abhängt, der in dem Eintrag variabler Länge 1416 des zweiten Abschnitts 1410B abgespeichert ist. Der Zeiger 1412B des Eintrags 1410 zeigt auf den Eintrag variabler Länge, 1416, des zweiten Abschnitts 1410B. Unter Verwendung von Verzeichniseinträgen fester Länge, 1412-1414 in dem ersten Abschnitt 1410A beschleunigt sich der Vorgang des Namen-Nachschauens. Zum Auffinden des nächsten Eintrags innerhalb eines Verzeichnisblocks 1410 ist keine Rechnung erforderlich. Durch Halten der Einträge 1412-1414 in dem ersten Abschnitt 1410A auf

einem kleinen Wert verbessert sich die Trefferrate für Dateisysteme mit einem Zeilenfüller-Datencache.

Meta-Daten

WAFL führt Information, die ein Dateisystem in Dateien beschreibt, welche als Meta-Daten bekannt sind. Meta-Daten umfassen eine Inodendatei, eine inomap-Datei und eine blkmap-Datei. WAFL speichert seine Meta-Daten in Dateien, die irgendwo auf einer Platte aufgezeichnet werden können. Weil sämtliche WAFL-Meta-Daten in Dateien geführt werden, lassen sie sich an eine beliebe Stelle schreiben, so wie jede andere Datei innerhalb der Datenbank.

Eine erste Metadaten-Datei ist die „Inodendatei", die Inoden enthält, welche sämtliche anderen Dateien innerhalb der Datenbank beschreiben. Figur 12 ist ein Diagramm einer Inodendatei 1210. Die Inodendatei 1210 kann irgendwo auf einer Platte aufgezeichnet werden, im Gegensatz zu bekannten Systemen, welche „Inodentabellen" auf eine feste Stelle der Platte schreiben. Die Inodendatei 1210 enthält eine Inode 1210A-1210F für jede Datei innerhalb des Dateisystems, ausgenommen die Inodendatei 1210 selbst. Gezeigt wird auf die Inodendatei 1210 durch eine als die „Wurzelinode" bezeichnete Inode. Die Wurzelinode wird an einer festen Stelle auf der Platte gehalten, bezeichnet als weiter unten noch zu beschreibender Dateisysteminformationsblock (fsinfo-Block). Die Inodendatei 1210 selbst ist in 4-KB-Blöcken auf der Platte (oder 4-KB-Puffern im Speicher) abgespeichert. Figur 12 veranschaulicht, daß Inoden 1210A-1210C in einem 4-KB-Puffer 1220 gespeichert sind. Für Größen von plattenintemen Inoden von 128 Bytes umfaßt ein 4-KB-Puffer (oder Block) 32 Inoden. Die Intern-Inodendatei 1210 setzt sich zusammen aus WAFL-Puffern 1220. Wenn eine Intern-Inode (das heißt 1210A) geladen wird, wird der plattenintemne Inodenteil der Intern-Inode 1210A für den Puffer 1220 der Inodendatei 1210 einkopiert. Die Pufferdaten selbst werden von der Platte her geladen. Das Schreiben von Daten auf die Platte erfolgt in umgekehrter Reihenfolge. Die Intern-Inode 1210A, die eine Kopie der plattenintemen Inode ist,

wird in den entsprechenden Puffer 1220 der Inodendatei 1210 kopiert. Anschließend wird die Inodendatei 1210 für das Schreiben zugewiesen, und die in dem Puffer 1220 der Inodendatei 1210 gespeicherten Daten werden auf die Platte geschrieben.

Eine weitere Metadaten-Datei ist die „Blockabbild"-Datei (blkmap-Datei). Figur 11A ist ein Diagramm, welches eine blkmap-Datei 1110 zeigt. Die blkmap-Datei 1110 enthält einen 32 Bits umfassenden Eintrag 1110A-1110C für jeden 4-KB-Block innerhalb des Plattenlaufwerksystems. Sie dient außerdem als Abbilddatei für freie Blöcke. Die blkmap-Datei 1110 gibt an, ob ein Plattenblock belegt wurde oder nicht. Figur 11B ist ein Diagramm eines Blockeintrags 1110A der blkmap-Datei 1110 (dargestellt in Figur 11A). Wie in Figur 11B gezeigt ist, umfaßt der Eintrag 1110A 32 Bits (BIT0-BIT31). Bit 0 (BIT0) des Eintrags 1110A ist das Aktiv-Dateisystem-Bit (FS-Bit). Das FS-Bit des Eintrags 1110A gibt an, ob der entsprechende Block Teil des aktiven Dateisystems ist oder nicht. Die Bits 1-20 (BIT1-BIT20) des Eintrags 1110A sind Bits, welche angeben, ob der Block Teil eines entsprechenden Schnappschusses (Zwischensicherung) 1-20 ist. Die nächsten oberen 10 Bits (BIT21-BIT30) sind reserviert. Bit 31 (BIT31) ist das Konsistenzpunkt-Bit (CP-BIT) des Eintrags 1110A.

Ein Block ist als ein freier Block in dem Dateisystem dann verfügbar, wenn sämtliche Bits (BIT0-BIT31) in dem 32 Bit umfassenden Eintrag 1110A für den Block gelöscht sind (auf einen Wert 0 zurückgesetzt). Figur 11C ist ein Diagramm, welches den Eintrag 1110A der Figur 11A veranschaulicht, wenn dieser anzeigt, daß der Plattenblock frei ist. Demnach ist der durch den Eintrag 1110A der blkmap-Datei 1110 referenzierte Block dann frei, wenn die Bits 0-31 (BIT0-BIT31) sämtlich einen Wert 0 haben. Figur 11D ist ein Diagramm, welches den Eintrag 1110A der Figur 11A in dem Zustand zeigt, in welchem er einen belegten Block in dem aktiven Dateisystem angibt. Wenn das Bit 0 (BIT0), auch als FS-Bit bezeichnet, auf einen Wert 1 gesetzt ist, kennzeichnet der Eintrag 1110A der blkmap-Datei 1110 einen Block, der Teil des aktiven Dateisystems ist. Bits 1-20 (BIT1-BIT20) dienen zum Anzeigen entsprechender Schnappschüsse, falls

vorhanden, die den Block referenzieren. Schnappschüsse werden unten im einzelnen erläutert. Wenn das Bit 0 (BIT0) auf einen Wert 0 gesetzt ist, so zeigt dies nicht unbedingt an, daß der Block für die Belegung zur Verfügung steht. Sämtliche Schnappschuß-Bits müssen 0 sein, damit der Block zugewiesen werden kann. Bit 31 (BIT31) des Eintrags 1110A hat stets denselben Zustand als Bit 0 (BIT0) auf der Platte, wird aber, wenn er in das Speicherbit 31 (BIT31) geladen wird, zur Buchführung als Teil eines Konsistenzpunkts verwendet.

Eine weitere Metadaten-Datei ist die „Inodenabbild"-Datei (inomap-Datei), die als ein Abbild für freie Inoden dient. Figur 13A ist ein Diagramm, welches eine Inodenabbild-Datei veranschaulicht. Die inomap-Datei 1310 enthält einen 8 Bits umfassenden Eintrag 1310A-1310C für jeden Block innerhalb der in Figur 12 gezeigten Inoden-Datei 1210. Jeder Eintrag 1310A-1310C ist eine Zählung zugeordneter oder belegter Inoden in dem entsprechenden Block innerhalb der Inoden-Datei 1210. Figur 13A zeigt Werte 32,5 bzw. 0 in den Einträgen 1310A-1310C. Die Inoden-Datei 1210 muß noch inspiziert werden, um herauszufinden, welche Inoden in dem Block frei sind, dies erfordert jedoch nicht das Umladen größerer Mengen beliebiger Blöcke von der Platte in den Speicher. Da jeder 4-KB-Block 1220 der Inodendatei 1210 32 Inoden aufnimmt, kann der 8 Bits umfassende inomap-Eintrag 1310A-1310C für jeden Block in der Inoden-Datei 1210 Werte annehmen, die zwischen 0 und 32 liegen. Wenn ein Block 1220 einer Inoden-Datei 1210 keine Inoden im Gebrauch hat, so ist der Eintrag 1310A-1310C für ihn innerhalb der Inomap-Datei 1310 „0". Wenn sämtliche Inoden in dem Block 1220 der Inodendatei 1210 im Gebrauch sind, hat der Eintrag 1310A-1310C der inomap-Datei 1310 einen Wert 32.

Figur 13B ist ein Diagramm, das eine inomap-Datei 1350 veranschaulicht, welche die 4-KB-Blöcke 1340A-1340C der Inoden-Datei 1340 referenziert. Beispielsweise speichert die Inoden-Datei 1340 37 Inoden in drei 4-KB-Blöcken 1340A-1340C. Blöcke 1340A-1340C der Inoden-Datei 1340 enthalten 32,5 bzw. 0 verwendete Inoden. Einträge 1350A-1350C der

blkmap-Datei 1350 referenzieren Blöcke 1340A-1340C der Inoden-Datei 1340. Damit haben die Einträge 1350A-1350C der inomap-Datei Werte von 32,5 und 0 für Blöcke 1340A-1340C der Inoden-Datei 1340. Die Einträge 1350A-1350C der inomap-Datei wiederum kennzeichnen 0,27 bzw. 32 freie Inoden in den Blöcken 1340A-1340C der Inoden-Datei 1340.

Bezugnehmend auf Figur 13 ist die Verwendung einer bitweisen Momentaufnahme für die Einträge 1310A-1310C der inomap-Datei 1310 an Stelle von Zählwerten deshalb von Nachteil, weil vier Bytes pro Eintrag 1310A-1310C für den Block 1220 der Inoden-Datei 1210 (in Figur 12 dargestellt), und nicht nur ein Byte erforderlich wären. Freie Inoden im Block bzw. in den Blöcken 1220 der Inoden-Datei 1210 müssen innerhalb der inomap-Datei 1310 deshalb nicht angezeigt werden, weil die Inoden selbst diese Information enthalten.

Figur 15 ist ein Diagramm, welches eine Dateisysteminformationsstruktur (fsinfo) 1510 veranschaulicht. Die Wurzelinode 1510B eines Dateisystems wird an einer festen Stelle auf der Platte gehalten, so daß sie beim Booten des Dateisystems geortet werden kann. Der fsinfo-Block ist keine Metaden-Datei, sondern Teil des WAFL-Systems. Die Wurzelinode 1510B ist eine Inode, die auf die Inoden-Datei 1210 Bezug nimmt. Sie ist Teil der Dateisysteminformationsstruktur (fsinfo) 1510, die außerdem Information 1510A einschließlich der Anzahl von Blöcken in dem Dateisystem, die Entstehungszeit des Dateisystems etc. enthält. Die vermischte Information 1510A enthält außerdem eine Prüfsumme 1510C (diese wird unten noch beschrieben). Mit Ausnahme der Wurzelinode 1510B selbst kann diese Information 1510A in einer Metadaten-Datei einer anderen Ausführungsform gehalten werden. In festen Plätzen auf der Platte werden zwei identische Kopien der fsinfo-Struktur 1510 gehalten.

Figur 16 ist ein Diagramm, welches das WAFL-Dateisystem 1670 in einem konsistenten Zustand auf einer Platte mit zwei fsinfo-Blöcken 1610 und 1612, einer Inoden-Datei 1620, einer blkmap-Datei 1630, einer inomap-Datei 1640, einem Wurzelverzeichnis 1650 und einer typischen Datei

(oder einem Verzeichnis) 1660 zeigt. Die Inoden-Datei 1620 besteht aus mehreren Inoden 1620A-1620D, welche andere Dateien 1630-1660 in dem Dateisystem 1670 referenzieren. Die Inode 1620A der Inoden-Datei 1620 referenziert die blkmap-Datei 1630. Die Inode 1620B referenziert die inomap-Datei 1640. Die Inode 1620C referenziert das Wurzelverzeichnis 1650. Die Inode 1620D referenziert eine typische Datei (oder ein typisches Verzeichnis) 1660. Somit zeigt die Inoden-Datei auf sämtliche Dateien 1630-1660 innerhalb des Dateisystems 1670, ausgenommen die fsinfo-Blöcke 1610 und 1612. Die fsinfo-Blöcke 1610 und 1612 enthalten jeweils eine Kopie 1610B bzw. 1612B der Inode der Inoden-Datei 1620. Weil die Wurzelinode 1610B und 1612B der fsinfo-Blöcke 1610 und 1612 die Inoden-Datei 1620 beschreibt, die ihrerseits den Rest der Dateien 1630-1660 in dem Dateisystem 1670 einschließlich sämtlicher Metadaten-Dateien 1630-1640 beschreibt, wird die Wurzelinode 1610B und 1612B als die Wurzel eines Baums von Blöcken betrachtet. Das WAFL-System 1620 verwendet diese Baumstruktur für ihr Aktualisierungsverfahren (Konsistenzpunkt) und zum Implementieren von Schnappschüssen, die beide unten noch beschrieben werden.

Liste von Inoden mit unsauberen Blöcken

Interne WAFL-Inoden (das heißt die WAFL-Inode 1010 gemäß Figur 10) des WAFL-Dateisystems werden in unterschiedlich verknüpften Listen entsprechend ihrem Status gehalten. Inoden, die sich auf unsaubere Blöcke beziehen, werden in einer in Figur 2 gezeigten Liste für unsaubere Inoden gehalten. Zulässige Daten enthaltende Inoden, die nicht unsauber sind, werden in einer separaten Liste gehalten, und Inoden, die keine zulässigen Daten aufweisen, werden in einer noch weiteren Liste geführt, wie dies im Stand der Technik bekannt ist. Die vorliegende Erfindung macht Gebrauch von einer Liste von Inoden mit unsauberen Datenblöcken, was das Auffinden sämtlicher Inoden erleichtert, bei denen Schreibzuweisungen erforderlich sind.

Figur 2 ist ein Diagramm, das eine Liste 210 unsauberer Inoden gemäß der Erfindung veranschaulicht. Die Liste 210 unsauberer Inoden enthält WAFL-interne Inoden 220-1750. Wie in Figur 17 gezeigt ist, enthält jede WAFL-interne Inode 220-250 einen Zeiger 220A-250A, der auf eine weitere Inode in der verknüpften Liste zeigt. Beispielsweise sind WAFL-Inoden 220-250 im Speicher an Stellen 2048, 2152, 2878, 3448 bzw. 3712 gespeichert. Dementsprechend enthält der Zeiger 220A der Inode 220 die Adresse 2152. Sie verweist deshalb auf die WAFL-Inode 222. Die WAFL-Inode 222 wiederum zeigt mit Hilfe der Adresse 2878 auf die WAFL-Inode 230. Die WAFL-Inode 230 verweist auf die WAFL-Inode 240. Die WAFL-Inode 240 zeigt auf die Inode 1750. Der Zeiger 250 der WAFL-Inode 250 enthält einen Null-Wert und zeigt daher nicht auf eine weitere Inode. Somit ist sie die letzte Inode innerhalb der Liste 210 für unsaubere Inoden. Jede Inode in der Liste 210 repräsentiert eine Datei aus einem Baum von Puffern, wie dies in Figur 10 dargestellt ist. Mindestens einer der von jeder Inode 220-250 referenzierte Puffer ist ein unsauberer Puffer. Ein unsauberer Puffer enthält modifizierte Daten, die auf eine neue Speicherplattenstelle in dem WAFL-System geschrieben werden müssen. WAFL schreibt stets unsaubere Puffer auf neue Speicherstellen der Platte.

KONSISTENZPUNKTE

Die WAFL-Plattenstruktur, wie sie bisher beschrieben wurde, ist statisch. Erfindungsgemäß werden Änderungen des Dateisystems 1670 streng gesteuert, um das Dateisystem 1670 in einem konsistenten Zustand zu halten. Das Dateisystem 1670 schreitet von einem selbstkonsistenten Zustand zu einem anderen selbstkonsistenten Zustand weiter. Die Menge (oder der Baum) selbstkonsistenter Blöcke auf der Platte mit ihrem Ursprung in der Wurzelinode 1510B wird als Konsistenzpunkt (CP) referenziert. Um Konsistenzpunkte zu implementieren, schreibt WAFL stets neue Daten in nicht-zugewiesene Blöcke auf der Platte. Es überschreibt niemals existierende Daten. Solange also die Wurzelinode 1510B nicht aktualisiert ist, ändert sich der Zustand des Dateisystems 1670, wie er sich auf der Platte darstellt, nicht. Damit das Dateisystem 1670 aber brauchbar ist, muß es

gelegentlich auf neu geschriebene Daten Bezug nehmen, und deshalb muß dann ein neuer Konsistenzpunkt geschrieben werden.

Bezugnehmend auf Figur 16, wird ein neuer Konsistenzpunkt dadurch geschrieben, daß zunächst sämtliche Dateisystem-Blöcke auf neue Stellen der Platten umgeräumt werden (einschließlich der Blöcke in Metadaten-Dateien, so wie die Inoden-Datei 1620, die blkmap-Datei 1630 und die inomap-Datei 1640). Eine neue Wurzelinode 1610B und 1612B für das Dateisystem 1670 wird dann auf die Platte geschrieben. Mit diesem Verfahren zur automatischen Aktualisierung eines Dateisystems ist das platteninterne Dateisystem niemals inkonsistent. Das platteninterne Dateisystem 1670 reflektiert einen alten Konsistenzpunkt, bis die Wurzelinode 1610B und 1612B geschrieben ist. Unmittelbar nach dem Schreiben der Wurzelinode 1610B und 1612B auf die Platte reflektiert das Dateisystem 1670 einen neuen Konsistenzpunkt. Datenstrukturen des Dateisystems 1670 können in beliebiger Reihenfolge aktualisiert werden, es gibt keinerlei Ordnungsbeschränkungen bei plattenintemen Schreibvorgängen, ausgenommen das eine Erfordernis, gemäß dem sämtliche Blöcke in dem Dateisystem 1670 auf die Platte geschrieben werden müssen, bevor die Wurzelinode 1610B und 1612B aktualisiert wird.

Um in einen neuen Konsistenzpunkt umgewandelt werden zu können, muß die Wurzelinode 1610B und 1612B zuverlässig und elementar aktualisiert werden. WAFL tut dies dadurch, daß zwei identische Kopien der fsinfo-Struktur 1610 und 1612 gehalten werden, welche die Wurzelinode 1610B und 1612B enthalten. Während der Aktualisierung der Wurzelinode 1610B und 1612B wird eine Kopie der fsinfo-Struktur 1610 auf die Platte geschrieben, anschließend wird die zweite Kopie der fsinfo-Struktur 1612 geschrieben. Eine Prüfsumme 1610C und 1612C in der fsinfo-Struktur 1610 bzw. 1612 dient zum Feststellen des Auftretens eines Systemzusammenbruchs, welches eine der Kopien der fsinfo-Struktur 1610 oder 1612, die jeweils eine Kopie der Wurzelinode enthalten, beim Schreiben auf die Platte verfälscht. Normalerweise sind die beiden fsinfo-Strukturen 1610 und 1612 identisch.

Algorithmus zum Erzeugen eines Konsistenzpunkts

Figur 5 ist ein Diagramm, welches das Verfahren zum Erzeugen eines Konsistenzpunkts veranschaulicht. Im Schritt 510 werden sämtliche „unsauberen" Inoden (also Inoden, die auf neue, modifizierte Daten enthaltende Blöcke zeigen) in dem System als im Konsistenzpunkt ihrer Inhalte befindlich markiert, und es wird nur ihr jeweiliger Inhalt auf die Platte geschrieben. Nur wenn diese Schreibvorgänge abgeschlossen sind, dürfen weitere Schreibvorgänge aus anderen Inoden die Platte erreichen. Außerdem können während der Zeit, in der unsaubere Schreibvorgänge stattfinden, keine neuen Modifikationen an Inoden vorgenommen werden, die sich in dem Konsistenzpunkt befinden.

Zusätzlich zur Einstellung des Konsistenzpunkt-Flags für sämtliche unsauberen Inoden, die Teil des Konsistenzpunkts sind, wird ein globales Konsistenzpunkt-Flag gesetzt, so daß seitens eines Benutzers angeforderte Änderungen sich in streng gesteuerter Weise verhalten. Nachdem das globale Konsistenzpunkt-Flag gesetzt ist, werden benutzerseitig angeforderte Änderungen, welche in dem Konsistenzpunkt befindliche Inoden beeinflussen, nicht zugelassen. Außerdem wird nur Inoden mit gesetztem Konsistenzpunkt-Flag Plattenspeicherplatz für ihre unsauberen Blöcke zugewiesen. Folglich wird der Zustand des Dateisystems auf die Platte geräumt, genauso, wie dies zu Beginn des Konsistenzpunkts geschah.

Im Schritt 520 werden reguläre Dateien auf Platte geräumt. Das Räumen regulärer Dateien umfaßt den Schritt des Zuweisens von Plattenspeicherplatz für unsaubere Blöcke in den regulären Dateien, außerdem das Schreiben der entsprechenden WAFL-Puffer auf die Platte. Die Inoden selbst werden anschließend in die Inoden-Datei geräumt (kopiert). Sämtliche Inoden, die zu beschreiben sind, befinden sich entweder in der Liste von Inoden mit unsauberen Puffern oder in der Liste von Inoden, die unsauber sind, jedoch keine unsauberen Puffer enthalten. Wenn der Schritt 520 abgeschlossen ist, gibt es keine weiteren regulären Inoden in dem Konsistenzpunkt, und sämtliche ankommenden E/A-Anforderungen verlaufen

erfolgreich, es sei denn, die Anforderungen verwenden Puffer, die für Platten-E/A-Operationen noch gesperrt sind.

Im Schritt 530 werden Spezialdateien auf die Platte geräumt. Das Räumen von Spezialdateien umfaßt den Schritt des Zuordnens von Plattenspeicherplatz für unsaubere Blöcke in den beiden Spezialdateien: die Inoden-Datei und die blkmap-Datei, das Aktualisieren des Konsistenzbits (CP-Bit), damit Übereinstimmung mit dem aktiven Dateisystem-Bit (FS-Bit) für jeden Eintrag in der blkmap-Datei herrscht, und anschließendes Einschreiben der Blöcke in die Platte. Die Schreibzuordnung der Inoden-Datei und der blkmap-Datei ist deshalb kompliziert, weil der Vorgang ihrer Schreibzuweisung die Dateien selbst ändert. Somit werden im Schritt 530 Schreibvorgänge gesperrt, während diese Dateien geändert werden, um zu verhindern, daß wichtige Blöcke für Platten-E/A-Operationen gesperrt werden, bevor die Änderungen abgeschlossen sind.

Im Schritt 530 werden außerdem die unten noch beschriebenen Schritte des Erzeugens und Löschens von Schnappschüssen durchgeführt, da dies der einzige zeitliche Punkt ist, zu welchem das Dateisystem – ausgenommen den fsinfo-Block – vollständig selbstkonsistent ist und gerade dabei ist, auf die Platte geschrieben zu werden. Ein Schnappschuß wird aus dem Dateisystem gelöscht, bevor ein neuer erzeugt wird, so daß in einem Durchgang dieselbe Schnappschuß-Inode verwendet werden kann.

Figur 6 ist ein Flußdiagramm, welches die Schritte darstellt, die der Schritt 530 umfaßt. Schritt 530 ordnet Plattenspeicherraum für die blkmap-Datei und die Inoden-Datei zu und kopiert das aktive FS-Bit in das CP-Bit für jeden Eintrag der blkmap-Datei. Dies garantiert, daß der Block in der Inoden-Datei, der die Inode der blkmap-Datei enthält, unsauber ist, so daß der Schritt 620 hierfür Plattenspeicherraum zuweist.

Im Schritt 620 wird für sämtliche unsauberen Blöcke in der Inode und den blkmap-Dateien Plattenspeicherplatz zugewiesen. Die unsauberen Blöcke

enthalten den Block der Inoden-Datei, der die Inode der blkmap-Datei als unsauberen Block enthält.

Im Schritt 630 wird die Inode für die blkmap-Datei erneut geräumt, allerdings wird diesmal die aktuelle Inode in den vorab geräumten Block in der Inoden-Datei geschrieben. Schritt 610 hat bereits den Block der Inoden-Datei verfälscht, welche die Inode der blkmap-Datei enthält. Damit braucht kein weiterer Schreibzuweisungsschritt entsprechend dem Schritt 620 geplant zu werden.

Im Schritt 640 werden die Einträge für jeden Block in der blkmap-Datei aktualisiert. Jeder Eintrag wird dadurch aktualisiert, daß das aktive FS-Bit in das CP-Bit kopiert wird (das heißt Einkopieren des Bits 0 in das Bit 31), und zwar bei sämtlichen Einträgen in unsauberen Blöcken innerhalb der blkmap-Datei.

Im Schritt 650 werden sämtliche unsauberen Blöcke in den blkmap- und Inoden-Dateien auf die Platte geschrieben.

Nur für Einträge in unsauberen Blöcken der blkmap-Datei muß das aktive Dateisystem-Bit (FS-Bit) im Schritt 640 in das Konsistenzpunkt-Bit (CP-Bit) kopiert werden. Unmittelbar nach einem Konsistenzpunkt besitzen sämtliche blkmap-Einträge denselben Wert sowohl für das aktive FS-Bit als auch das CP-Bit. Mit fortschreitender Zeit werden einige aktive FS-Bits von blkmap-Datei-Einträgen für das Dateisystem entweder gelöscht oder gesetzt. Die Blöcke der blkmap-Datei, die geänderte FS-Bits enthalten, werden entsprechend als unsauber markiert. Während des folgenden Konsistenzpunkts brauchen saubere Blöcke nicht zurückkopiert zu werden. Die sauberen Blöcke werden deshalb nicht kopiert, sie an dem vorhergehenden Konsistenzpunkt nicht unsauber waren und sich in den Blöcken seitdem nichts geändert hat. Solange also das Dateisystem zu Beginn mit dem aktiven FS-Bit und dem CP-Bit gleichen Werts in sämtlichen blkmap-Einträgen erzeugt wurde, brauchen lediglich Einträge bei unsauberen Blöcken in jedem Konsistenzpunkt aktualisiert zu werden.

Bezugnehmend auf Figur 5 wird im Schritt 540 der Dateisysteminformationsblock (Fsinfo) aktualisiert und dann auf die Platte geräumt. Der Fsinfo-Block wird dadurch aktualisiert, daß in ihn für die Inoden-Datei eine neue Wurzelinode eingeschrieben wird. Der Fsinfo-Block wird zweimal geschrieben. Zuerst wird er an eine Stelle und dann an eine zweite Stelle geschrieben. Die zwei Schreibvorgänge werden derart ausgeführt, daß dann, wenn während des einen oder des anderen Schreibvorgangs ein Systemzusammenbruch erfolgt, auf der Platte ein selbstkonsistentes Dateisystem vorliegt. Bei einem Systemzusammenbruch während des Schreibvorgangs des zweiten Fsinfo-Blocks ist dann entweder der neue Konsistenzpunkt verfügbar, oder es ist der vorhergehende Konsistenzpunkt (auf der Platte vor Beginn des jüngsten Konsistenzpunkts) vorhanden, wenn der erste Fsinfo-Block ausgefallen ist. Wenn das Dateisystem nach einem Systemausfall neu gestartet wird, wird die höchste Generationenzählung für einen Konsistenzpunkt in den Fsinfo-Blöcken mit einem korrekten Prüfsummenwert verwendet. Dies wird weiter unten noch näher erläutert.

Im Schritt 550 wird der Konsistenzpunkt abgeschlossen. Dies macht es erforderlich, daß jegliche unsaubere Inoden, die, weil sie nicht Teil des Konsistenzpunkts waren, neu in die Warteschlange gestellt werden. Sämtliche Dioden, die ihren Zustand während des Konsistenzpunkts geändert haben, werden in die Konsistenzpunkt-Warteschlange (CP_WAIT) gestellt. Die CP_WAIT-Warteschlange enthält Inoden, die sich vor Abschluß des Schritts 540 geändert haben, jedoch nach dem Schritt 510, wenn der Konsistenzpunkt gestartet ist. Nach Abschluß des Konsistenzpunkts werden die Inoden in der CP_WAIT-Warteschlange neu eingeordnet, entsprechend der regulären Liste von Dioden mit unsauberen Puffern und der Liste von unsauberen Inoden ohne unsaubere Puffer.

<u>Einzelordnungsbeschränkung des Konsistenzpunkts</u>

Wie in den Figuren 20A-20C dargestellt ist, besitzt die vorliegende Erfindung eine Einzelordnungsbeschränkung. Die Einzelordnungsbeschränkung besagt, daß der Fsinfo-Block 1810 nur auf Platte geschrieben wird, nach-

dem sämtliche übrigen Blöcke auf die Platte geschrieben sind. Das Schreiben des Fsinfo-Blocks 1810 ist elementar, weil ansonsten das gesamte Dateisystem 1830 verloren gehen könnte. Damit erfordert das WAFL-Dateisystem, daß der Fsinfo-Block 1810 auf einmal geschrieben wird und sich nicht in einem inkonsistenten Zustand befindet. Wie in Figur 15 gezeigt ist, enthält jeder der Fsinfo-Blöcke 1810 (1510) eine Prüfsumme 1510C und eine Generationenzählung 1510D.

Figur 20A veranschaulicht das Aktualisieren der Generationenzählung 1810D und 1870D der Fsinfo-Blöcke 1810 und 1870. Jedesmal, wenn ein Konsistenzpunkt (oder Schnappschuß) ausgeführt wird, wird auch die Generationenzählung des Fsinfo-Blocks aktualisiert. Figur 20A zeigt zwei Fsinfo-Blöcke 1810 und 1870 mit Generationenzählungen 1810D und 1870D, die den gleichen Wert N aufweisen, was einen Konsistenzpunkt für das Dateisystem angibt. Beide Fsinfo-Blöcke referenzieren den vorausgehenden Konsistenzpunkt (das alte Dateisystem auf der Platte) 1830. Eine neue Version des Dateisystems existiert auf der Platte und wird als neuer Konsistenzpunkt 1831 referenziert. Die Generationenzählung wird bei jedem Konsistenzpunkt erhöht.

In Figur 20B wird die Generationenzählung 1810D des ersten Fsinfo-Blocks 1810 aktualisiert und erhält einen Wert N+1. Dann wird sie auf die Platte geschrieben. Figur 20B veranschaulicht einen Wert N+1 für die Generationenzählung 1810D des Fsinfo-Blocks 1810, wohingegen die Generationenzählung 1870D des zweiten Fsinfo-Blocks 1870 einen Wert von N hat. Der Fsinfo-Block 1810 referenziert den neuen Konsistenzpunkt 1831, wohingegen der Fsinfo-Block 1870 den alten Konsistenzpunkt 1830 referenziert. Als nächstes wird die Generationenzählung 1870D des Fsinfo-Blocks 1870 aktualisiert und auf Platte geschrieben, wie dies in Figur 20C dargestellt ist. In Figur 20C besitzt die Generationenzählung 1870D des Fsinfo-Blocks 1870 einen Wert N+1. Deshalb besitzen beide Fsinfo-Blöcke 1810 und 1870 den gleichen Generationen-Zählerstand N+1.

Kommt es zu einem Systemzusammenbruch zwischen zwei Fsinfo-Block-Aktualisierungen, besitzt jede Kopie des Fsinfo-Blocks 1810 und 1870 eine (in dem Diagramm nicht gezeigte) selbstkonsistente Prüfsumme, jedoch weist eine der Generationenzahlen 1810D oder 1870D einen höheren Wert auf. Ein Systemzusammenbruch geschieht, wenn das Dateisystem sich in dem in Figur 20B gezeigten Zustand befindet. In der bevorzugten Ausführungsform der vorliegenden Erfindung gemäß Figur 20B wird die Generationenzählung 1810D des Fsinfo-Blocks 1810 vor dem zweiten Fsinfo-Block 1870D aktualisiert. Daher ist die Generationenzählung 1810D (mit dem Wert Eins) größer als die Generationenzählung 1870D des Fsinfo-Blocks 1870. Da die Generationenzählung des ersten Fsinfo-Blocks 1810 größer ist, wird sie zur Wiederherstellung des Dateisystems nach einem Systemzusammenbruch ausgewählt. Dies geschieht deshalb, weil der erste Fsinfo-Block 1810 mehr laufende Daten enthält, was durch seine Generationenzählung 1810D angegeben wird. Falls der erste Fsinfo-Block verfälscht wird, da bei seiner Aktualisierung das System zusammenbricht, so wird die andere Kopie 1870 des Fsinfo-Blocks zur Wiederherstellung des Dateisystems 1830 in konsistentem Zustand verwendet.

Erfindungsgemäß ist es nicht möglich, beide Fsinfo-Blöcke 1810 und 1870 gleichzeitig zu aktualisieren. Deshalb existiert in dem Dateisystem mindestens eine gute Kopie des Fsinfo-Blocks 1810 und 1870. Dies macht es möglich, das Dateisystem stets in einem konsistenten Zustand wiederherzustellen.

WAFL macht keine speziellen Wiederherstellungsprozeduren erforderlich. Dies unterscheidet es von bekannten Systemen, die von Protokollierung, geordneten Schreibvorgängen und streng geordneten Schreibvorgängen bei der Wiederherstellung Gebrauch machen. Dies deshalb, weil nur Datenverfälschung, gegen die RAID Schutz bietet, oder Software ein WAFL-Dateisystem verfälschen kann. Um Datenverlust bei einem Systemausfall zu vermeiden, kann WAFL ein nicht-flüchtiges Transaktions-Protokoll für sämtliche Operationen führen, die nach dem jüngsten Konsistenzpunkt erfolgt sind. Dieses Protokoll ist völlig unabhängig vom WAFL-

Plattenformat und ist nur erforderlich, um zu verhindern, daß bei einem Systemzusammenbruch Operationen verlorengehen. Allerdings ist es nicht erforderlich, die Konsistenz des Dateisystems beizubehalten.

Erzeugen eines Konsistenzpunkts

Wie oben beschrieben, werden Änderungen des WAFL-Dateisystems streng gesteuert, um das Dateisystem in einem konsistenten Zustand zu halten. Figuren 17A-17H veranschaulichen die Erzeugung eines Konsistenzpunkts für ein WAFL-Dateisystem. Die Erzeugung eines Konsistenzpunkts wird anhand der Figuren 5 und 6 erläutert.

In den Figuren 17A-17L sind Puffer, die nicht modifiziert wurden, ohne Sternchen neben sich. Deshalb enthalten Puffer die gleichen Daten wie entsprechende platteninterne Blöcke. Damit läßt sich ein Block in den Speicher laden, er ist gegenüber seiner platteninternen Version jedoch unverändert. Ein Puffer mit einem einzelnen Sternchen (*) daneben bedeutet einen unsauberen Puffer in dem Speicher (seine Daten sind modifiziert). Ein Puffer mit einem doppelten Sternchen (**) neben sich bedeutet einen unsauberen Puffer, dem Plattenspeicherplatz zugewiesen ist. Schließlich ist ein Puffer mit einem Dreifachsternchen (***) ein unsauberer Puffer, der in einen neuen Block auf der Platte eingeschrieben ist. Die Konvention zum Bezeichnen des Zustands von Puffern wird auch bei den Figuren 21A-21E benutzt.

Figur 17A zeigt eine Liste 2390 von Inoden mit unsauberen Puffern, umfassend Inoden 2306A und 2306B. Die Inoden 2306A und 2306B referenzieren Bäume von Puffern, in denen mindestens ein Puffer jedes Baums modifiziert wurde. Zu Beginn werden Konsistenzpunkt-Flags 2391 und 2392 der Inoden 2306A und 2306B gelöscht (0). Während für das vorliegende System eine Liste 2390 von Inoden mit unsauberen Puffern dargestellt ist, sollte dem Fachmann ersichtlich sein, daß andere Listen von Inoden ebenfalls im Speicher existieren können. Beispielsweise wird in dem Speicher eine Liste von Dioden geführt, die unsauber sind, allerdings keine

unsauberen Puffer haben. Diese Inoden müssen als in dem Konsistenzpunkt befindlich markiert werden. Sie müssen auf die Platte geräumt werden, damit auch der unsaubere Inhalt der Inoden-Datei auf die Platte geschrieben wird, selbst wenn unsaubere Inoden nicht unsaubere Blöcke referenzieren. Dies geschieht im Schritt 520 in Figur 5.

Figur 17B ist ein Diagramm, welches ein WAFL-Dateisystem eines vorhergehenden Konsistenzpunkts mit dem Fsinfo-Block 2302, der Inoden-Datei 2346, der blkmap-Datei 2344 sowie Dateien 2340 und 2342 umfaßt. Die Datei 2340 enthält Blöcke 2310-2314, die Daten „A“, „B“ bzw. „C“ enthalten. Die Datei 2342 enthält Datenblöcke 2316-2320 mit Daten „D“, „E“ bzw. „F“. Die blkmap-Datei 2344 enthält den Block 2324. Die Inoden-Datei 2346 enthält zwei 4 KB-Blöcke 2304 und 2306. Der zweite Block 2306 enthält Inoden 2306A-2306C, die die Datei 2340, die Datei 2342 bzw. die blkmap-Datei 2344 referenzieren. Dies ist im Block 2306 durch Auflistung der Dateinummer in der Diode angezeigt. Fsinfo-Block 2302 enthält die Wurzelinode. Die Wurzelinode referenziert die Blöcke 2304 und 2306 der Inoden-Datei 2346. Figur 17B veranschaulicht einen Baum von Puffern in einem Dateisystem mit Wurzelbildung durch den Fsinfo-Block 2302, welcher die Wurzelinode beinhaltet.

Figur 17C ist ein Diagramm welches zwei modifizierte Puffer für die Blöcke 2314 und 2322 im Speicher veranschaulicht. Das aktive Dateisystem wird so modifiziert, daß der die Daten „C“ enthaltende Block 2314 aus der Datei 2340 gelöscht wird. Außerdem werden die im Block 2320 gespeicherten Daten „F“ zu „F-Prime“ modifiziert und in einem Puffer für den Plattenblock 2322 gespeichert. Es sollte gesehen werden, daß die in Puffern für Plattenblöcke 2314 und 2322 enthaltene modifizierte Daten zu dieser Zeit nur im Speicher existieren. Sämtliche übrigen Blöcke in dem aktiven Dateisystem der Figur 17C sind nicht modifiziert und deshalb nicht mit einem Sternchen neben ihnen markiert. Allerdings können einige oder sämtliche dieser Blöcke in dem Speicher zugehörige saubere Puffer aufweisen.

Figur 17D ist ein Diagramm, welches die Einträge 2324A-2324M der blkmap-Datei 2344 im Speicher veranschaulicht. Einträge 2324A-2324M sind in einem Puffer für den 4-KB-Block 2324 der blkmap-Datei 2344 enthalten. Wie zuvor beschrieben, sind das BIT0 und BIT31 das FS-BIT bzw. das CP-BIT. Das Konsistenzpunkt-Bit (CP-BIT) wird während eines Konsistenzpunkts gesetzt, um zu garantieren, daß der entsprechende Block nach Beginn, jedoch noch nicht erfolgtem Abschluß eines Konsistenzpunkts modifiziert wird. BIT1 ist das erste Schnappschuß-Bit (wird unten beschrieben). Blkmap-Einträge 2324A und 2324B veranschaulichen, daß gemäß Figur 17B die 4-KB-Blöcke 2304 und 2306 der Inoden-Datei 2346 in dem aktiven Dateisystem (FS-BIT gleicht 1) und in dem Konsistenzpunkt (CP-BIT gleicht 1) sind. In ähnlicher Weise sind die übrigen Blöcke 2310-2312 und 2316-2320 sowie 2324 in dem aktiven Dateisystem und in dem Konsistenzpunkt. Allerdings sind die Blöcke 2308 und 2322 sowie 2326-2328 weder in dem aktiven Dateisystem noch in dem Konsistenzpunkt (was durch BIT0 bzw. BIT31 angegeben wird). Der Eintrag für den gelöschten Block 2314 hat einen Wert 0 im FS-BIT, was anzeigt, daß er aus dem aktiven Dateisystem entfernt wurde.

Im Schritt 510 der Figur 5 werden sämtliche „unsauberen" Inoden in dem System als im Konsistenzpunkt befindlich markiert. Unsaubere Inoden enthalten sowohl Inoden, die unsauber sind, als auch Inoden, welche unsaubere Puffer referenzieren. Figur 17I veranschaulicht eine Liste von Inoden mit unsauberen Puffern, wo die Konsistenzpunkt-Flags 2391 und 2392 von Inoden 2306A und 2306B gesetzt (1) sind. Die Inode 2306A referenziert den Block 2314, der Daten „C" der Datei 2340 enthält, die aus dem aktiven Dateisystem zu löschen ist. Die Inode 2306B des Blocks 2306 der Inoden-Datei 2346 referenziert die Datei 2342. Der Block 2320, der die Daten „F" enthält, wurde modifiziert, und es muß ein neuer Block zugewiesen werden, der die Daten „F" enthält. Im Schritt 510 werden die unsauberen Inoden 2306A und 2306B in den Puffer für den Block 2308 einkopiert. Der Puffer für den Block 2306 wird anschließend (im Schritt 530) auf Platte geschrieben. Dies ist in Figur 17E dargestellt. Die modifizierten Daten existieren nur in dem Speicher, und der Puffer 2308 ist als unsauber

markiert. Die Inkonsistenzpunkt-Flags 2391 und 2392 der Inoden 2306A und 2306B werden anschließend gelöscht (0), wie in Figur 17A dargestellt. Dies gibt die Inoden für die Benutzung durch andere Prozesse frei.

Im Schritt 520 werden reguläre Dateien auf Platte geräumt. Damit wird dem Block 2322 Plattenspeicherplatz zugewiesen. Der Block 2314 der Datei 2340 ist zu löschen, so daß mit diesem Block nichts geschieht, bis später dann der Konsistenzpunkt abgeschlossen ist. Der Block 2322 wird im Schritt 520 auf Platte geschrieben. Dies ist in Figur 17F dargestellt, wo Puffer für die Blöcke 2322 und 2314 auf Platte geschrieben wurden (markiert durch ***). Die Zwischen-Zuordnung von Plattenspeicherraum (**) ist nicht dargestellt. Die Inoden 2308A und 2308B des Blocks 2308 der Inoden-Datei 2346 werden anschließend in die Inoden-Datei geräumt. Die Inode 2308A des Blocks 2308 referenziert Blöcke 2310 und 2312 der Datei 2346. Die Inode 2308B referenziert Blöcke 2316, 2318, 2322 für die Datei 2342. Wie in Figur 17F gezeigt ist, wird Plattenspeicherplatz für den Block 2308 der Inode 2346 und für den direkten Block 2322 der Datei 2342 zugewiesen. Allerdings ist das Dateisystem selbst noch nicht aktualisiert worden. Damit bleibt das Dateisystem in einem konsistenten Zustand.

Im Schritt 530 wird die blkmap-Datei 2344 auf Platte geräumt. Dies ist in Figur 17G dargestellt, wo die blkmap-Datei 2344 durch ein Sternchen als unsauber gekennzeichnet ist.

Im Schritt 610 der Figur 6 wird die Inode für die blkmap-Datei vorab in die Inoden-Datei geräumt, wie in Figur 17H gezeigt. Die Inode 2308C wurde in den Block 230B der Inoden-Datei 2346 geräumt. Allerdings referenziert die Inode 2308C immer noch den Block 2324. Im Schritt 620 wird Plattenspeicherraum für die blkmap-Datei 2344 und die Inoden-Datei 2346 zugewiesen. Der Block 2308 wird für die Inoden-Datei 2346 zugewiesen, und Block 2326 wird für die blkmap-Datei 2344 zugewiesen. Wie oben beschrieben, enthält der Block 2308 der Inoden-Datei 2346 eine vorabgeräumte Inode 2308C für die blkmap-Datei 2344. Im Schritt 630 wird die Inode für die blkmap-Datei 2344 in den vorgeräumten Block 2308C in der

Inode 2346 geschrieben. Damit wird im Schritt 620 die interne Inode 2308C zum Referenzieren des Blocks 2324 aktualisiert und wird in den Puffer des Speichers kopiert, der den in den Block 2308 zu schreibenden Block 2306 enthält. Dies ist in Figur 17H dargestellt, wo die Inode 2308C den Block 2326 referenziert.

Im Schritt 640 werden die Einträge 2326A-2326L für jeden Block 2304-2326 in der blkmap-Datei 2344 in Figur 17J aktualisiert. Blöcke, die sich nach dem Beginn des Konsistenzpunkts in Figur 17B nicht geändert haben, besitzen in ihren Einträgen dieselben Werte. Die Einträge werden dadurch aktualisiert, daß BIT0 (das FS-Bit) in das Konsistenzpunkt-Bit (BIT31) kopiert wird. Der Block 2306 ist nicht Teil des aktiven Dateisystems, und deshalb ist BIT0 gleich Null (BIT0 wurde im Schritt 620 ausgeschaltet, als der Block 2308 zugewiesen wurde, um neue Daten für diesen Teil der Inoden-Datei aufzunehmen). Dies ist in Figur 17J für den Eintrag 2326B dargestellt. In ähnlicher Weise ist im Eintrag 2326F für den Block 2314 der Datei 2340 das BIT0 und das BIT31 gleich Null. Block 2320 der Datei 2342 und Block 2324 der blkmap-Datei 2344 werden in ähnlicher Weise gehandhabt, wie dies für die Einträge 2361 bzw. 2326K gezeigt ist. Im Schritt 650 werden der unsaubere Block 2308 der Inoden-Datei 2346 und der unsaubere Block 2326 der blkmap-Datei 2344 auf Platte geschrieben. Dies ist in Figur 17K durch ein dreifaches Sternchen (***) neben den Blöcken 2308 und 2326 angegeben.

Bezugnehmend auf Figur 5 wird im Schritt 540 der Dateisysteminformationsblock 2302 auf Platte geräumt, und dies geschieht zweimal. Damit ist der Fsinfo-Block 2302 unsauber geworden und wird anschließend auf Platte geschrieben (in Figur 17L durch ein Dreifachsternchen angedeutet). In Figur 17L ist ein einzelner Fsinfo-Block 2302 dargestellt. Wie aus dem Diagramm ersichtlich ist, referenziert der Fsinfo-Block 2302 jetzt den Block 2304 und den Block 2308 der Inoden-Datei 2346. In Figur 17L ist der Block 2306 nicht mehr Bestandteil der Inoden-Datei 2346 des aktiven Dateisystems. In ähnlicher Weise enthält die durch die Inode 2308A der Inoden-Datei 2346 referenzierte Datei 2340 Blöcke 2310 und 2312. Der

Block 2314 ist nicht mehr Bestandteil der Datei 2340 innerhalb dieses Konsistenzpunkts. Die Datei 2342 enthält Blöcke 2316, 2318 und 2322 in dem neuen Konsistenzpunkt, während Block 2320 nicht Bestandteil der Datei 2342 ist. Weiterhin referenziert der Block 2308 der Inoden-Datei 2346 eine neue blkmap-Datei 2344 mit dem Block 2326.

Wie in Figur 17L gezeigt ist, wird in einem Konsistenzpunkt das aktive Dateisystem dadurch aktualisiert, daß die Inode der Inoden-Datei 2346 in den Fsinfo-Block 2302 einkopiert wird. Allerdings verbleiben die Blöcke 2314, 2320, 2324 und 2306 des vorhergehenden Konsistenzpunkts auf der Platte. Diese Blöcke werden beim Aktualisieren des Dateisystems niemals überschrieben, um zu garantieren, daß sowohl der alte Konsistenzpunkt 1830 als auch der neue Konsistenzpunkt 1831 auf der Platte vorhanden sind, siehe Figur 20 und Schritt 540.

Schnappschüsse

Das WAFL-System arbeitet mit Schnappschüssen. Ein Schnappschuß oder eine Momentaufnahme ist eine nur lesbare Kopie eines gesamten Dateisystems zu einem gegebenen Augenblick, zu welchem der Schnappschuß erzeugt wird. Ein neu erzeugter Schnappschuß bezieht sich auf exakt dieselben Plattenblöcke, wie dies das aktive Dateisystem tut. Deshalb wird er innerhalb einer kurzen Zeitspanne erzeugt und verbraucht keinen zusätzlichen Plattenspeicherplatz. Nur wenn Datenblöcke innerhalb des aktiven Dateisystems modifiziert und in neue Stellen auf der Platte geschrieben werden, beginnt der Schnappschuß, besonderen Platz zu beanspruchen.

WAFL hält bis zu 20 unterschiedliche Schnappschüsse, die von 1 bis 20 numeriert sind. Damit ermöglicht WAFL die Erzeugung mehrfacher „Klone“ desselben Dateisystems. Jeder Schnappschuß wird durch eine Schnappschuß-Inode repräsentiert, die ähnlich der Darstellung des aktiven Dateisystems durch eine Wurzelinode ist. Schnappschüsse werden erzeugt durch Duplizieren der Wurzeldatenstruktur des Dateisystems. In der bevorzugten Ausführungsform ist die Wurzeldatenstruktur die Wurzelinode.

Allerdings könnte auch jede andere Datenstruktur verwendet werden, die repräsentativ für ein gesamtes Dateisystem ist. Die Schnappschuß-Inoden befinden sich an einer festen Stelle innerhalb der Inoden-Datei. Die Begrenzung auf 20 Schnappschüsse wird durch die Größe der Blockabbild-Einträge bestimmt. WAFL erfordert zwei Schritte zum Erzeugen eines neuen Schnappschusses N: Kopieren der Wurzelanode in die Anode für den Schnappschuß N und Kopieren des Bits 0 in das Bit N jedes Blockabbild-Eintrags innerhalb der blkmap-Datei. Bit0 gibt die Blöcke an, die von dem Baum unterhalb der Wurzelinode referenziert werden.

Das Ergebnis ist ein neuer Dateisystembaum, dessen Wurzel gebildet wird durch die Schnappschuß-Inode N, die exakt dieselben Plattenblöcke referenziert wie die Wurzelinode. Durch Einstellen eines entsprechenden Bits in der Blockabbildung für jeden Block in dem Schnappschuß wird verhindert, daß Schnappschuß-Blöcke freigesetzt werden, selbst wenn die aktive Datei die Schnappschuß-Blöcke nicht mehr verwendet. Da WAFL stets neue Daten auf unbenutzte Speicherplätze schreibt, ändert sich der Schnappschuß-Baum selbst dann nicht, wenn das aktive Dateisystem sich ändert. Da ein neu erzeugter Schnappschuß-Baum exakt die gleichen Blöcke wie die Wurzelinode referenziert, verbraucht er keinen zusätzlichen Plattenspeicherplatz. Im Lauf der Zeit referenziert der Schnappschuß Plattenblöcke, die ansonsten freigesetzt würden. Damit benutzen im Verlauf der Zeit der Schnappschuß und das aktive Dateisystem immer weniger Blöcke, so daß der von dem Schnappschuß beanspruchte Raum zunimmt. Schnappschüsse können gelöscht werden, wenn sie eine nicht mehr akzeptierbare Anzahl von Plattenblöcken belegen.

Die Liste aktiver Schnappschüsse wird zusammen mit den Namen der Schnappschüsse in einer Schnappschuß-Verzeichnis genannten Metadaten-Datei abgespeichert. Der Plattenzustand wird in der oben beschriebenen Weise aktualisiert. Wie bei sämtlichen anderen Änderungen erfolgt die Aktualisierung durch automatisches Weiterschreiten von einem Konsistenzpunkt zum anderen. Modifizierte Blöcke werden in unbenutzte Plätze

auf der Platte geschrieben, woraufhin eine neue Wurzelinode, welche das aktualisierte Dateisystem beschreibt, geschrieben wird.

Überblick über Schnappschüsse

Figur 18A ist ein Diagramm des Dateisystems 1830, bevor ein Schnappschuß aufgenommen wird, wobei Umwege-Ebenen entfernt wurden, um einen einfacheren Überblick über das WAFL-Dateisystem zu ermöglichen. Das Dateisystem 1830 repräsentiert das in Figur 16 gezeigte Dateisystem 1690. Das Dateisystem 1830 besteht aus Blöcken 1812 bis 1820. Die Inode der Inoden-Datei ist in dem Fsinfo-Block 1810 enthalten. Während eine einzelne Kopie des Fsinfo-Blocks 1810 in Figur 18A dargestellt ist, versteht sich natürlich, daß auf der Platte eine zweite Kopie des Fsinfo-Blocks vorhanden ist. Die in dem Fsinfo-Block 1810 enthaltene Inode 1810A enthält 16 Zeiger, die auf 16 Blöcke mit gleicher Umwegeebene zeigen. Die Blöcke 1810-1820 in Figur 18A repräsentieren sämtliche Blöcke innerhalb des Dateisystems 1830 einschließlich direkte Blöcke, indirekte Blöcke, etc. Obschon lediglich fünf Blöcke 1812-1820 dargestellt sind, kann jeder Block auf weitere Blöcke verweisen.

Figur 18B ist ein Diagramm, das die Erzeugung eines Schnappschusses zeigen. Der Schnappschuß wird für das gesamte Dateisystem 1830 dadurch erstellt, daß einfach die Inode 1810A der Inoden-Datei kopiert wird, die in dem Fsinfo-Block 1810 gespeichert ist, wobei die Inode in die Schnappschuß-Inode 1822 einkopiert wird. Durch Einkopieren der Inode 1810A der Inoden-Datei wird eine neue Datei von Dioden erzeugt, die das gleiche Dateisystem wie das aktive Dateisystem repräsentiert, weil die Inode 1810A der Inoden-Datei selbst kopiert wird. Es brauchen keine weiteren Blöcke 1812-1820 dupliziert zu werden. Die kopierte Inode oder Schnappschuß-Inode 1822 wird dann in die Inoden-Datei einkopiert, was einen Block innerhalb der Inoden-Datei unsauber macht. Für eine Inoden-Datei aus einer oder mehreren Umwegeebenen wird jeder indirekte Block wiederum unsauber gemacht. Dieser Vorgang des Verunreinigens von Blökken schreitet durch sämtliche Umwegeebenen. Jeder 4-KB-Block inner-

halb der Inoden-Datei auf der Platte enthält 32 Inoden, wo jede Inode 128 Bytes Länge aufweist.

Die neue Schnappschuß-Inode 1822 nach Figur 18B verweist zurück auf die Blöcke 1812-1820 höchster Umwegeebene, referenziert durch die Inode 1810A der Inoden-Datei, wenn der Schnappschuß 1822 aufgenommen wird. Die Inoden-Datei selbst ist eine rekursive Struktur, weil sie Schnappschüsse des Dateisystems 1830 beinhaltet. Jeder Schnappschuß 1822 ist eine Kopie der Inode 1810A der Inoden-Datei, welche in die Inoden-Datei einkopiert wird.

Figur 18C ist ein Diagramm, das das aktive Dateisystem 1830 und den Schnappschuß 1822 für den Zeitpunkt veranschaulicht, zu dem eine Änderung des aktiven Dateisystems 1830 nach Aufnahme des Schnappschusses 1822 stattfindet. Wie in dem Diagramm gezeigt, wird der Block 1818 mit den Daten „D“ nach Aufnahme des Schnappschusses (Figur 18B) modifiziert, und deshalb wird ein neuer Block 1824 mit Daten „D_{prime}“ für das aktive Dateisystem 1830 zugeordnet. Damit enthält das aktive Dateisystem 1830 Blöcke 1812-1816 und 1820-1824, es enthält aber nicht den Block 1818 mit den Daten „D“. Allerdings wird der die Daten „D“ enthaltende Block 1818 deshalb nicht überschrieben, weil das WAFL-System keine Blöcke auf der Platte überschreibt. Der Block 1818 wird gegen ein Überschreiben von einem Schnappschuß-Bit geschützt, welches in dem Blockabbild-Eintrag für den Block 1818 gesetzt wird. Deshalb zeigt der Schnappschuß 1822 immer noch auf den unmodifizierten Block 1818 ebenso wie auf die Blöcke 1812-1816 und 1820. Die vorliegende Erfindung unterscheidet sich gemäß den Figuren 18A-18C von bekannten Systemen, die „Klone“ eines Dateisystems erzeugen, wobei ein Klon eine Kopie sämtlicher Blöcke einer Diodendatei auf einer Platte ist. Damit werden die gesamten Inhalte der herkömmlichen Inoden-Dateien dupliziert, was große Mengen (MB) an Plattenspeicherplatz ebenso erfordert wie beträchtliche Zeit für Platten-E/A-Operationen.

Wenn das aktive Dateisystem 1830 in Figur 18C modifiziert wird, benötigt es deshalb mehr Plattenspeicherraum, weil das Dateisystem mit den Blökken 1812-1820 nicht überschrieben wird. In Figur 18C ist der Block 1818 als ein direkter Block dargestellt. Bei einem wirklichen Dateisystem allerdings kann der Block 1818 auch durch einen indirekten Block mittels Zeiger angesprochen werden. Wenn also der Block 1818 modifiziert und an einen neuer Stelle der Platte als Block 124 abgespeichert wird, werden auch die entsprechenden direkten und indirekten Blöcke kopiert und dem aktiven Dateisystem 1830 zugeordnet.

Figur 19 ist ein Diagramm, welches die Änderungen veranschaulicht, die im Block 1824 gemäß Figur 18C auftreten. Der Block 1824 nach Figur 18C ist in der gestrichelten Linie 1824 in Figur 19 dargestellt. Figur 19 veranschaulicht verschiedene Umwegeebenen für den Block 1824 nach Figur 18C. Der neue Block 1910, welcher gemäß Figur 18C auf die Platte geschrieben wird, ist in Figur 19 mit 1910 bezeichnet. Weil der Block 1824 einen Datenblock 1910 enthält, welcher modifizierte Daten beinhaltet, die durch einen doppelten Umweg oder doppelten Verweis referenziert werden, werden auch zwei weitere Blöcke 1918 und 1926 modifiziert. Der Zeiger 1924 eines einfach-indirekten Blocks 1918 referenziert einen neuen Block 1910, und deshalb muß der Block 1918 an eine neue Stelle der Platte geschrieben werden. In ähnlicher Weise wird der Zeiger 1928 des indirekten Blocks 1926 modifiziert, da er auf den Block 1918 zeigt. Deshalb kann gemäß Figur 19 das Modifizieren eines Datenblocks 1910 zur Folge haben, daß mehrere indirekte Blöcke 1918 und 1926 ebenfalls modifiziert werden. Dies macht es erforderlich, auch die Blöcke 1918 und 1926 auf eine neue Stelle der Platte zu schreiben.

Da die direkten und indirekten Blöcke 1910, 1918 und 1926 des Datenblocks 1824 in Figur 18C geändert und an eine neue Stelle geschrieben wurden, wird die Inode in der Inoden-Datei in einen neuen Block geschrieben. Der modifizierte Block der Inoden-Datei erhält einen neuen Block auf der Platte, da Daten nicht überschrieben werden können.

Wie in Figur 19 gezeigt ist, wird auf den Block 1910 durch indirekte Blökke 1926 bzw. 1918 gezeigt. Wenn also der Block 1910 modifiziert und an einer neuen Stelle der Platte gespeichert wird, werden auch die entsprechenden direkten und indirekten Blöcke kopiert und dem aktiven Dateisystem zugeordnet. Damit muß eine Reihe von Datenstrukturen aktualisiert werden. Das Ändern des direkten Blocks 1910 und der indirekten Blöcke 1918 und 1926 veranlaßt, daß die blkmap-Datei modifiziert werden muß.

Die Schlüsseldatenstrukturen für Schnappschüsse sind die Blockabbild-Einträge, wo jeder Eintrag mehrere Bits für einen Schnappschuß aufweist. Dies ermöglicht es, daß mehrere Schnappschüsse erzeugt werden. Ein Schnappschuß ist ein Bild eines Baums von Blöcken, die das Dateisystem (1830 in Figur 18) bilden. Solange keine neuen Daten auf Blöcke des Schnappschusses geschrieben werden, wird das durch den Schnappschuß repräsentierte Dateisystem nicht geändert. Ein Schnappschuß ist einem Konsistenzpunkt ähnlich.

Das erfindungsgemäße Dateisystem ist vollständig konsistent nach dem letzten Mal des Schreibens der Fsinfo-Blöcke 1810 und 1870. Wenn daher das System einen Netzausfall erleidet, entsteht beim Neustart das Dateisystem 1830 in konsistentem Zustand. Da 8-32 MB Plattenspeicherraum bei einem typischen bekannten „Klon“ eines 1-GB-Dateisystems verwendet werden, führen Klone nicht zu Konsistenzpunkten oder Schnappschüssen wie die vorliegende Erfindung.

Bezugnehmend auf Figur 22 existieren zwei frühere Schnappschüsse 2110A und 2110B auf der Platte. Zu dem Zeitpunkt, zu dem ein dritter Schnappschuß entsteht, wird die auf das aktive Dateisystem zeigende Wurzelinode in den Inodeneintrag 2110C für den dritten Schnappschuß in der Inoden-Datei 2110 kopiert. Gleichzeitig zeigt in dem durchgehenden Konsistenzpunkt ein Flag an, daß der Schnappschuß 3 erzeugt wird. Das gesamte Dateisystem wird verarbeitet, indem geprüft wird, ob BIT0 für jeden Eintrag innerhalb der blkmap-Datei gesetzt (1) oder gelöscht (0) ist. Sämtliche BIT0-Werte für jeden Blockabbild-Eintrag werden in die Ebene

für den Schnappschuß 3 kopiert. Nach Beendigung ist jeder aktive Block 2110-2116 und 1207 in dem Dateisystem zu diesem Zeitpunkt in dem Schnappschuß aufgenommen.

Blöcke, die durchgängig für eine gegebene Zeitspanne auf der Platte existiert haben, befinden sich ebenfalls in den entsprechenden Schnappschüssen 2110A-2110B, die dem dritten Schnappschuß 2110C vorausgehen. Wenn ein Block in dem Dateisystem für eine ausreichend lange Zeitspanne verblieben ist, ist er in sämtlichen Schnappschüssen enthalten. Der Block 1207 ist ein derartiger Block. Wie in Figur 22 gezeigt ist, wird der Block 1207 durch die Inode 2210G der aktiven Inoden-Datei referenziert, außerdem indirekt durch die Schnappschüsse 1, 2 und 3.

Die sequentielle Reihenfolge von Schnappschüssen repräsentiert nicht unbedingt eine chronologische Reihenfolge von Dateisystem-Kopien. Jeder einzelne Schnappschuß in einem Dateisystem kann zu jeder gegebenen Zeit gelöscht werden, um dadurch einen Eintrag für nachfolgenden Gebrauch verfügbar zu machen. Wenn BIT0 eines blkmap-Eintrags, der das aktive Dateisystem referenziert, gelöscht wird (was bedeutet, daß der Block aus dem aktiven Dateisystem gelöscht wurde), so kann der Block nicht noch einmal benutzt werden, wenn irgendeines der Schnappschuß-Referenzbits gesetzt wird. Dies deshalb, weil der Block Teil eines Schnappschusses ist, der noch in Gebrauch ist. Ein Block kann nur neu verwendet werden, wenn sämtliche Bits in dem blkmap-Eintrag auf Null gesetzt sind.

Algorithmus zum Erzeugen eines Schnappschusses

Das Erzeugen eines Schnappschusses entspricht etwa exakt der Erzeugung eines regulären Konsistenzpunkts gemäß Figur 5. Im Schritt 510 werden sämtliche unsauberen Inoden als in dem Konsistenzpunkt befindlich markiert. Im Schritt 520 werden reguläre Dateien auf die Platte geräumt. Im Schritt 520 werden Spezialdateien (das heißt die Inoden-Datei und die blkmap-Datei) auf Platte geräumt. Im Schritt 540 werden Fsinfo-Blöcke

auf Platte geräumt. Im Schritt 550 werden sämtliche Inoden, die sich nicht im Konsistenzpunkt befanden, verarbeitet. Figur 5 wird oben im einzelnen beschrieben. Tatsächlich erfolgt das Erzeugen eines Schnappschusses als Teil der Erzeugung eines Konsistenzpunkts. Der Hauptunterschied zwischen der Erzeugung eines Schnappschusses und der eines Konsistenzpunkts besteht darin, daß sämtliche Einträge der blkmap-Datei das aktive FS-Bit in das Schnappschuß-Bit einkopiert haben. Das Schnappschuß-Bit repräsentiert den entsprechenden Schnappschuß, um die Blöcke in dem Schnappschuß gegen Überschreiben zu schützen. Das Erzeugen und das Löschen von Schnappschüssen erfolgt im Schritt 530, da dies der einzige Punkt ist, an dem das Dateisystem vollständig selbst konsistent ist und auf dem Wege zur Platte ist.

Im Schritt 530 werden unterschiedliche Schritte durchgeführt, die dann in Figur 6 dargestellt sind, und zwar für einen Konsistenzpunkt, wenn ein neuer Schnappschuß erzeugt wird. Die Schritte sind sehr ähnlich jenen für einen regulären Konsistenzpunkt. Figur 7 ist ein Flußdiagramm, welches die Schritte zeigt, welche der Schritt 530 zum Erzeugen eines Schnappschusses umfaßt. Wie oben beschrieben, weist der Schritt 530 Plattenspeicherplatz für die blkmap-Datei und die Inoden-Datei zu und kopiert das aktive FS-Bit in das Schnappschuß-Bit, welches den entsprechenden Schnappschuß repräsentiert, um die Blöcke in dem Schnappschuß gegen Überschreiben zu schützen.

Im Schritt 710 werden die Inoden der blkmap-Datei und des Schnappschusses auf Platte vorgeräumt. Zusätzlich zu dem Räumen der Inode und der blkmap-Datei in einen Block der Inoden-Datei (wie im Schritt 610 der Figur 6 für einen Konsistenzpunkt), wird die Inode des erzeugten Schnappschusses auch in einen Block der Inoden-Datei geräumt. Dies garantiert, daß der Block in der Inoden-Datei, der die Inode des Schnappschusses enthält, unsauber ist.

Im Schritt 720 wird jeder Block in der blkmap-Datei unsauber gemacht. Im Schritt 760 (unten beschrieben) werden sämtliche Einträge in der blkmap-

Datei aktualisiert, und nicht nur die Einträge in unsauberen Blöcken. Somit müssen sämtliche Blöcke der blkmap-Datei hier als unsauber markiert werden, um zu garantieren, daß der Schritt 730 für sie Plattenspeicherplatz zum Schreiben zuweist.

Im Schritt 730 wird für sämtliche unsauberen Blöcke in der Inode und in blkmap-Dateien Plattenspeicherplatz zugewiesen. Die unsauberen Blöcke enthalten den Block in der Inoden-Datei, welche die Inode der blkmap-Datei, die unsauber ist, enthält, außerdem den Block, der die Inode für den neuen Schnappschuß enthält.

Im Schritt 740 werden die Inhalte der Wurzelinode für das Dateisystem in die Inode des Schnappschusses innerhalb der Inoden-Datei kopiert. Zu dieser Zeit wird jedem Block, der Teil des neuen Konsistenzpunkts ist und der auf Platte geschrieben wird, Plattenspeicherplatz zugeordnet. Damit kopiert ein Duplizieren der Wurzelinode in die Schnappschuß-Inode in wirksamer Weise das gesamte aktive Dateisystem. Die aktuellen Blöcke, die in dem Schnappschuß enthalten sind, sind die gleichen Blöcke des aktiven Dateisystems.

Im Schritt 750 werden die Inoden der blkmap-Datei und der Schnappschuß in die Inoden-Datei kopiert.

Im Schritt 760 werden Einträge in der blkmap-Datei aktualisiert. Zusätzlich zu dem Kopieren des aktiven FS-Bits in das CP-Bit für die Einträge wird das aktive FS-Bit auch in das dem neuen Schnappschuß entsprechende Schnappschuß-Bit kopiert.

Im Schritt 770 werden sämtliche unsauberen Blöcke in den blkmap- und Inoden-Dateien auf Platte geschrieben.

Schließlich werden zu einer gewissen Zeit Schnappschüsse selbst aus dem Dateisystem entfernt, Schritt 760. Ein Schnappschuß wird dadurch aus dem Dateisystem entfernt, daß sein Schnappschuß-Inodeneintrag innerhalb

der Inoden-Datei des aktiven Dateisystems gelöscht und jedes Bit, das der Schnappschußnummer in jedem Eintrag innerhalb der blkmap-Datei entspricht, gelöscht wird. Es erfolgt eine Zählung auch für jedes Bit des Schnappschusses in sämtlichen blkmap-Einträgen, die aus einem eingestellten Wert gelöscht werden, um dadurch eine Zählung der durch Löschen des Schnappschusses freigesetzten Blöcke zu schaffen (entsprechend der freigesetzten Menge an Plattenspeicherplatz). Das System entscheidet anhand des ältesten Schnappschusses, welcher Schnappschuß gelöscht werden soll. Auch Benutzer können von Hand spezifizierte Schnappschüsse löschen.

Die vorliegende Erfindung begrenzt die Gesamtanzahl von Schnappschüssen und führt eine blkmap-Datei, die Einträge mit Mehrfach-Bits zum Verfolgen der Schnappschüsse anstelle der Verwendung von Zeigern mit einem COW-Bit, wie dies in Episode der Fall ist, aufweist. Ein nicht verwendeter Block enthält für sämtliche Bits in seinem blkmap-Datei-Eintrag nur Nullen. Im Verlauf der Zeit wird das BIT0 für das aktive Dateisystem üblicherweise zu einem gegebenen Zeitpunkt eingeschaltet. Das Setzen des BIT0 identifiziert den entsprechenden Block als in dem aktiven Dateisystem zugeordnet. Wie oben angegeben, werden sämtliche Schnappschuß-Bits zu Beginn auf Null gesetzt. Wenn das aktive Dateibit vor Setzen irgendeines Schnappschuß-Bits gelöscht ist, ist der Block in keinem auf Platte gespeicherten Schnappschuß vorhanden. Deshalb steht der Block sofort zur Neuzuweisung zur Verfügung und kann später aus einem Schnappschuß nicht wiedergewonnen werden.

Erzeugung eines Schnappschusses

Wie oben beschrieben, ist ein Schnappschuß einem Konsistenzpunkt sehr ähnlich. Deshalb soll die Erzeugung eines Schnappschusses unter Bezugnahme auf die Unterschiede zwischen ihr und der Erzeugung eines Konsistenzpunkts gemäß Figuren 17A-17L erläutert werden. Figuren 21A-21F zeigen die Unterschiede bei der Erzeugung eines Schnappschusses.

Figuren 17A-17D zeigen den Zustand des WAFL-Dateisystems, wenn ein Schnappschuß begonnen wird. Sämtliche unsauberen Inoden werden als in dem Konsistenzpunkt befindlich markiert, Schritt 510, und im Schritt 520 werden die regulären Dateien auf Platte geräumt. Damit ist die Anfangsverarbeitung für einen Schnappschuß identisch mit der eines Konsistenzpunkts. Die Verarbeitung für einen Schnappschuß unterscheidet sich im Schritt 530 von der des Konsistenzpunkts. Im folgenden wird die Verarbeitung eines Schnappschusses gemäß Figur 7 erläutert.

Die folgende Beschreibung gilt für einen zweiten Schnappschuß des WAFL-Dateisystems. Ein erster Schnappschuß ist in den blkmap-Einträgen der Figur 17C aufgezeichnet. Wie in den Einträgen 2324A-2324M, den Blöcken 2304-2306, 2310-2320 und 2324 dargestellt, sind diese in dem ersten Schnappschuß enthalten. Sämtliche anderen Schnappschuß-Bits (BIT1-BIT20) haben angenommener Weise den Wert 0, was anzeigt, daß ein entsprechender Schnappschuß auf der Platte nicht vorliegt. Figur 21A zeigt das Dateisystem nach Abschluß der Schritte 510 und 520.

Im Schritt 710 werden Inoden 2308C und 2308D des Schnappschusses 2 und der blkmap-Datei 2344 auf Platte geräumt. Dies stellt sicher, daß der Block der Inoden-Datei, der die Schnappschuß-2-Inode enthalten wird, unsauber ist. In Figur 21B werden Inoden 2308C und 2308D für den Schnappschuß 2 und für die blkmap-Datei 2344 vorgeräumt.

Im Schritt 720 ist die gesamte blkmap-Datei 2344 unsauber gemacht. Dies veranlaßt die gesamte blkmap-Datei 2344, im Schritt 730 Plattenraum zugewiesen zu bekommen. Im Schritt 730 wird Plattenraum für unsaubere Blöcke 2308 und 2326 für die Inoden-Datei 2346 und die blkmap-Datei 2344 gemäß Figur 21C zugewiesen. Angedeutet ist dies durch ein Dreifachsternchen (***) neben den Blöcken 2308 und 2326. Dies unterscheidet sich von der Erzeugung eines Konsistenzpunkts, bei dem Plattenspeicherplatz nur für Blöcke zugewiesen ist, deren Einträge sich im Schritt 620 der Figur 6 innerhalb der blkmap-Datei 2344 geändert haben. Die blkmap-Datei 2344 nach Figur 21C enthält einen einzelnen Block 2324. Wenn al-

lerdings die blkmap-Datei 2344 mehr als einen Block umfaßt, so wird im Schritt 730 Plattenspeicherplatz für sämtliche Blöcke zugewiesen.

Im Schritt 740 wird die Wurzelinode für das neue Dateisystem in die Inode 2308D für Schnappschuß 2 kopiert. Im Schritt 750 werden die Inoden 2308C und 2308D der blkmap-Datei 2344 und der Schnappschuß 2 auf Platte geräumt, wie in Figur 21D gezeigt ist. Das Diagramm veranschaulicht, daß die Schnappschuß-2-Inode 2308D Blöcke 2304 und 2308, nicht aber Block 2306 referenziert.

Im Schritt 760 werden Einträge 2326A-2326L im Block 2326 der blkmap-Datei 2344 gemäß Figur 21E aktualisiert. Das Diagramm zeigt, daß das Schnappschuß-2-Bit (BIT2) ebenso wie das FS-BIT und das CP-BIT für jeden Eintrag 2326A-2326L aktualisiert wird. Damit sind die Blöcke 2304, 2308-2312, 2316-2318, 2322 und 2326 im Schnappschuß 2 enthalten, die Blöcke 2306, 2314, 2320 und 2324 jedoch nicht. Im Schritt 770 werden die unsauberen Blöcke 2308 und 2326 auf Platte geschrieben.

Die weitere Verarbeitung des Schnappschusses 2 ist identisch mit der Erzeugung eines Konsistenzpunkts, wie dies in Figur 5 gezeigt ist. Im Schritt 540 werden zwei Fsinfo-Blöcke auf Platte geräumt. Figur 21F repräsentiert das WAFL-Dateisystem in einem konsistenten Zustand anschließend an diesen Schritt. Die Dateien 2340, 2342, 2344 und 2346 des konsistenten Dateisystems nach Abschluß des Schritts 540 sind durch gestrichelte Linien in Figur 21F angegeben. Im Schritt 550 wird der Konsistenzpunkt durch Verarbeitung der Inoden, die nicht in dem Konsistenzpunkt waren, abgeschlossen.

Zugriffszeit-Überschreibungen

Unix-Dateisysteme müssen in jeder Inode eine „Zugriffszeit" (atime von access time) enthalten. Atime gibt den letzten Zeitpunkt des Lesens der Datei an. Er wird jedesmal aktualisiert, wenn ein Zugriff auf die Datei erfolgt. Wenn also eine Datei gelesen wird, wird der Block, der die Inode in

der Inoden-Datei enthält, neu geschrieben, um die Inode zu aktualisieren. Dies könnte von Nachteil für die Erzeugung von Schnappschüssen deshalb sein, weil als Konsequenz das Lesen einer Datei möglicherweise Speicherplatz auf der Platte benötigt. Außerdem könnte das Lesen sämtlicher Dateien innerhalb des Dateisystems zur Folge haben, daß die gesamte Inoden-Datei dupliziert wird. Die vorliegende Erfindung löst dieses Problem.

Wegen des Vorhandenseins von Atime könnte ein Lesevorgang möglicherweise Plattenspeicherplatz verbrauchen, da ein Modifizieren einer Inode zur Folge hat, daß ein neuer Block für die Inoden-Datei auf die Platte geschrieben wird. Außerdem könnte ein Lesevorgang möglicherweise fehlschlagen, wenn ein Dateisystem voll ist, demzufolge ein abnormaler Zustand des Dateisystems auftritt.

Im allgemeinen werden Daten auf einer Platte in dem WAFL-Dateisystem nicht überschrieben, um auf der Platte gespeicherte Daten zu schützen. Die einzige Ausnahme dieser Regel besteht darin, daß Atime für eine Inode überschreibt, wie dies in den Figuren 23A-23B gezeigt ist. Wenn ein „Atime-Überschreiben“ stattfindet, bestehen die einzigen in einem Block der Inoden-Datei modifizierten Daten in Atime für eine oder mehrere der Inoden, die sie enthält, und der Block wird an derselben Stelle neu geschrieben. Dies ist die einzige Ausnahme innerhalb des WAFL-Systems. Im übrigen werden neue Daten stets auf neue Plattenspeicherplätze geschrieben.

In Figur 23A sind die Atimes 2423 und 2433 einer Inode 2422 in einem alten WAFL-Inoden-Datei-Block 2420 und die Schnappschußinode 2432, die den Block 2420 referenziert, dargestellt. Die Inode 2422 des Blocks 2420 referenziert direkt den Block 2410. Atime 2423 der Inode 2422 ist „4/30 9:15 PM“, während Atime 2433 der Schnappschuß-Inode 2432 „5/1 10:00 AM“ ist. Figur 23A veranschaulicht das Dateisystem vor einem Zugriff auf den direkten Puffer 2410.

Figur 23B veranschaulicht die Inode 2422 des direkten Blocks 2410, nachdem auf den direkten Block 2410 zugegriffen wurde. Wie in dem Diagramm dargestellt, wird die Zugriffszeit 2423 der Inode 2422 mit der Zugriffszeit 2433 des Schnappschusses 2432, den sie referenziert, überschrieben. Damit wird die Zugriffszeit 2423 der Inode 2422 für den direkten Block 2410 „5/1 11:23 AM".

Das Zulassen des Überschreibens von Inoden-Datei-Blöcken mit neuen Zugriffszeiten (Atime) führt zu einer leichten Inkonsistenz innerhalb des Schnappschusses. Die Atime für eine Datei in einem Schnappschuß kann tatsächlich später liegen als der Zeitpunkt, zu dem der Schnappschuß erzeugt wurde. Um Benutzer an einem Feststellen dieser Inkonsistenz zu hindern, justiert WAFL den Wert Atime für sämtliche Dateien innerhalb eines Schnappschusses auf diejenige Zeit ein, zu der der Schnappschuß tatsächlich erzeugt wurde, und nicht auf die Zeit, zu der auf die Datei zuletzt zugegriffen wurde. Diese Schnappschußzeit wird in der Inode gespeichert, die den Schnappschuß in seiner Gesamtheit beschreibt. Wenn folglich über den Schnappschuß zugegriffen wird, so wird die Zugriffszeit 2423 für die Inode 2422 stets in Form „5/1 10:00 AM" gemeldet. Dies geschieht sowohl vor dem Aktualisieren, wenn man „4/30 9:15 PM" erwarten könnte, als auch nach der Aktualisierung, wenn „5/1 11:23 AM" erwartet werden könnte. Erfolgt ein Zugriff durch das aktive Dateisystem, so werden die Zeiten in der Form „4/30 9:15 PM" und „5/1 11:23 AM" vor bzw. nach dem Aktualisieren gemeldet. Auf diese Weise wird ein Verfahren zum Führen eines Dateisystems in einem konsistenten Zustand und zum Erzeugen von ausschließlich lesbaren Kopien des Dateisystems offenbart.

Patentansprüche

1. Verfahren zum Erzeugen eines Konsistenzpunkts, umfassend die Schritte:

Markieren (510) einer Mehrzahl von Inoden, wobei eine Inode eine Dateidefinitionsstruktur ist, die zumindest eine Datei in einem Dateisystem beschreibt, die auf mehrere modifizierte Blöcke in einem Dateisystem verweist, als in einem Konsistenzpunkt befindlich;

Räumen (520) regulärer Dateien sowie Metadateien (530) auf eine Speichereinrichtung;

Räumen (540) mindestens eines Blocks von Dateisysteminformation auf die Speichereinrichtung; und

erneutes Einstellen (550) jeglicher berührter Inoden, die nicht Teil des Konsistenzpunkts waren, in eine Warteschlange.

2. Verfahren nach Anspruch 1, bei dem der Schritt des Räumens von Metadateien auf die Speichereinrichtung weiterhin folgende Schritte beinhaltet:

Vorräumen (610) einer Inode aus einer Blockabbildungsdatei in eine Inodendatei;

Zuweisen (620) von Platz auf der Speichereinrichtung für sämtliche berührten Blöcke in der Inode und den Blockabbildungsdateien;

erneutes Räumen (630) der Inode für die Blockabbildungsdatei;

Aktualisieren (640) einer Mehrzahl von Einträgen in der Blockabbildungsdatei, wobei jeder Eintrag unter den mehreren Einträgen einen Block auf der Speichereinrichtung repräsentiert; und

Schreiben (650) sämtlicher berührter Blöcke in der Blockabbildungsdatei und der Inodendatei auf die Speichereinrichtung.

0702815

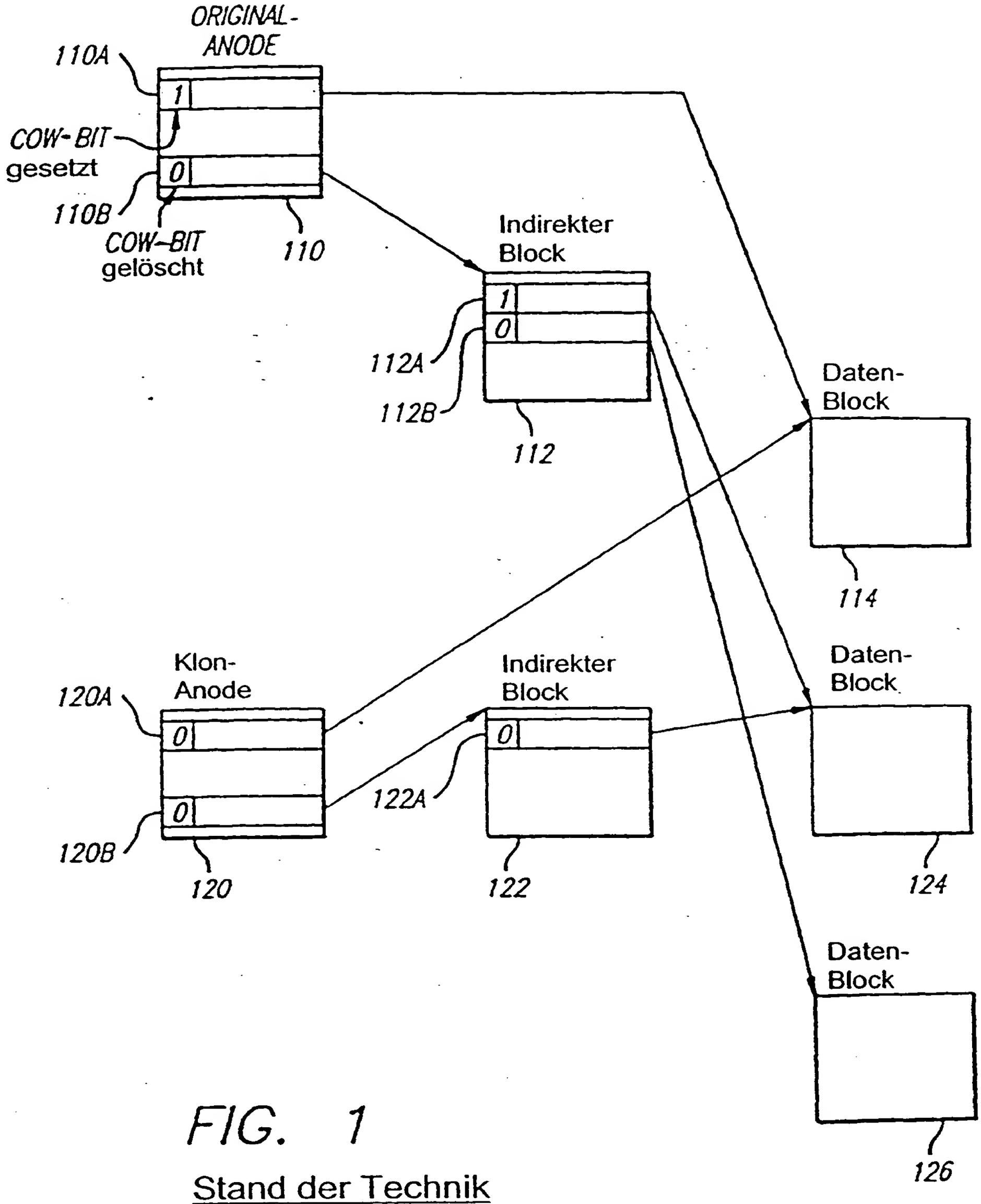


FIG. 1

Stand der Technik

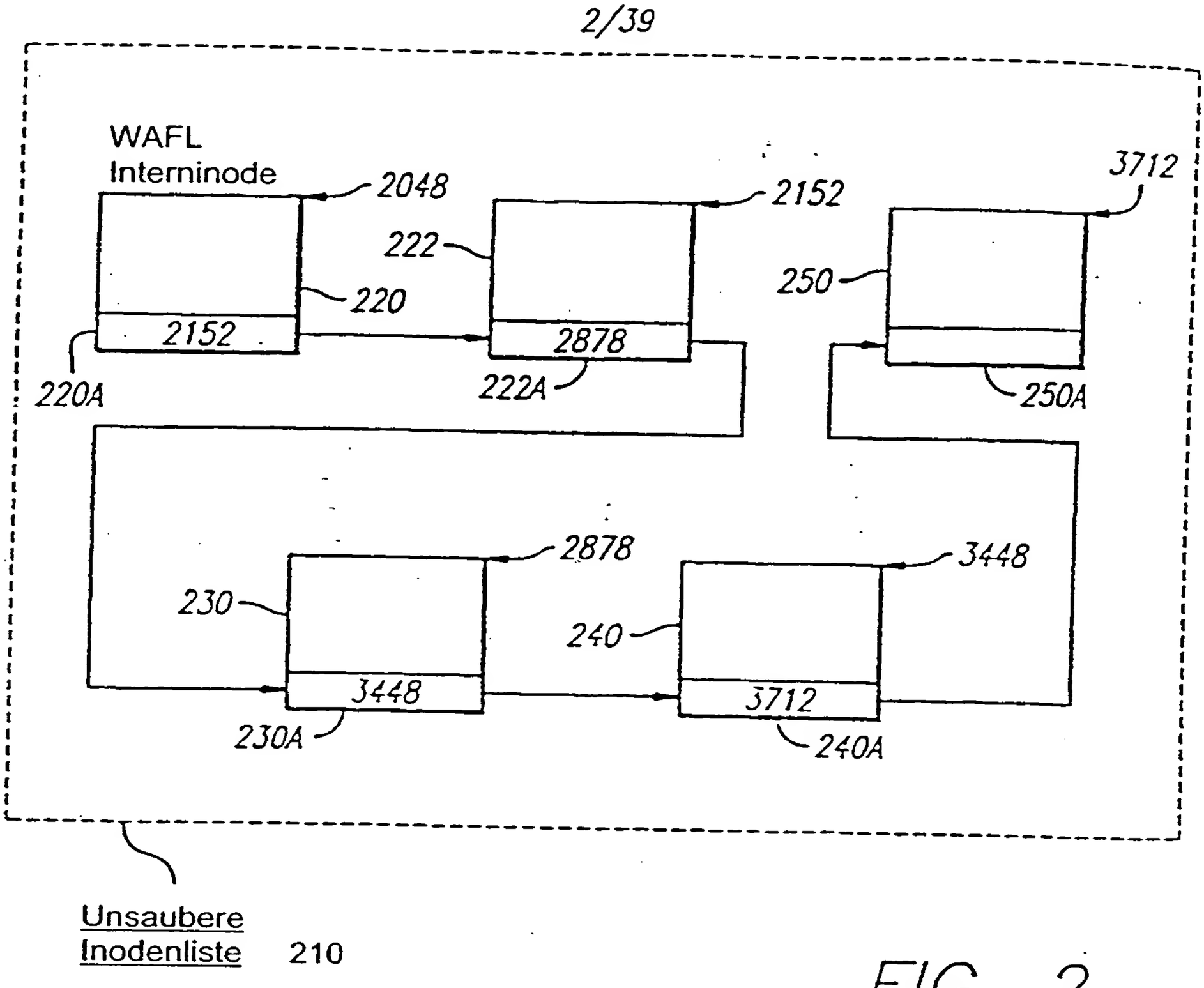

WAFL
Interninode
2048
2152
222
220
2152
2878
220A
222A
3712
250
250A
2878
3448
230
240
3448
3712
230A
240A
Unsaubere
Inodenliste 210

FIG. 2

FIG. 3

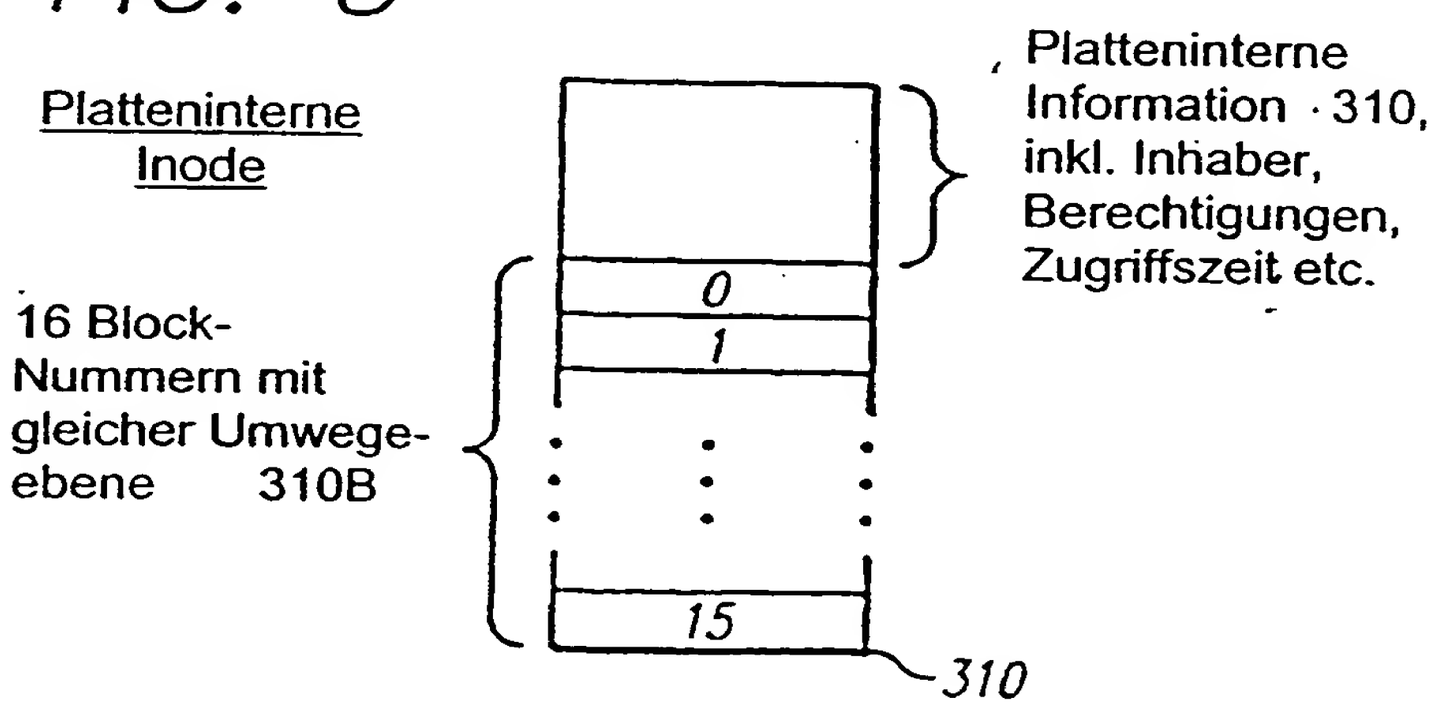

Platteninterne
Inode
16 Block-
Nummern mit
gleicher Umwege-
ebene 310B
Platteninterne
Information · 310,
inkl. Inhaber,
Berechtigungen,
Zugriffszeit etc.
0
1
15
310

FIG. 4A

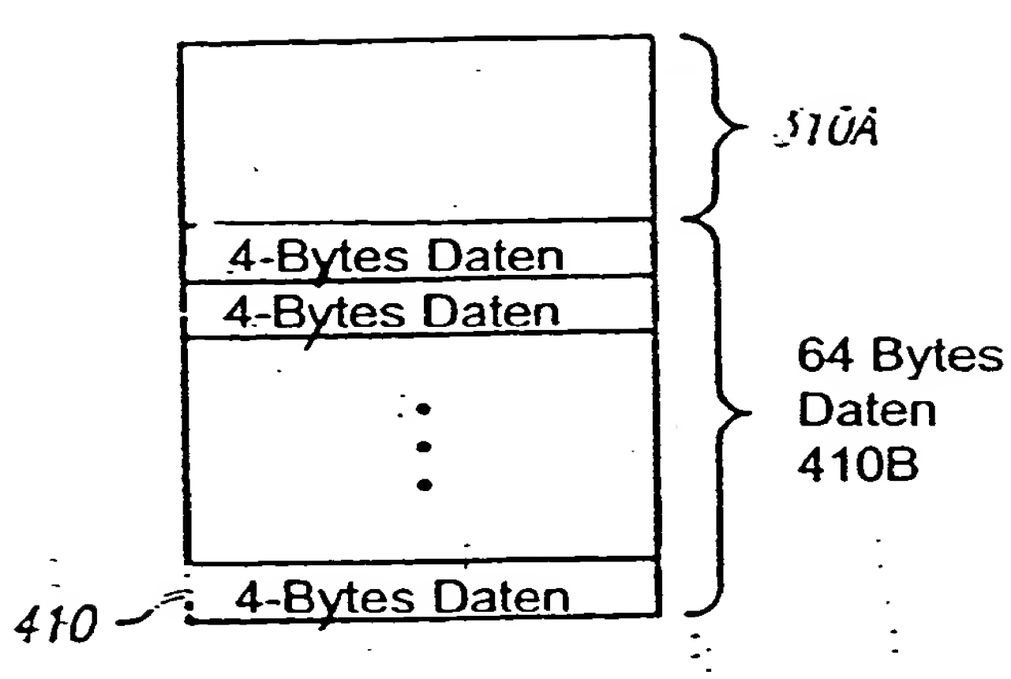
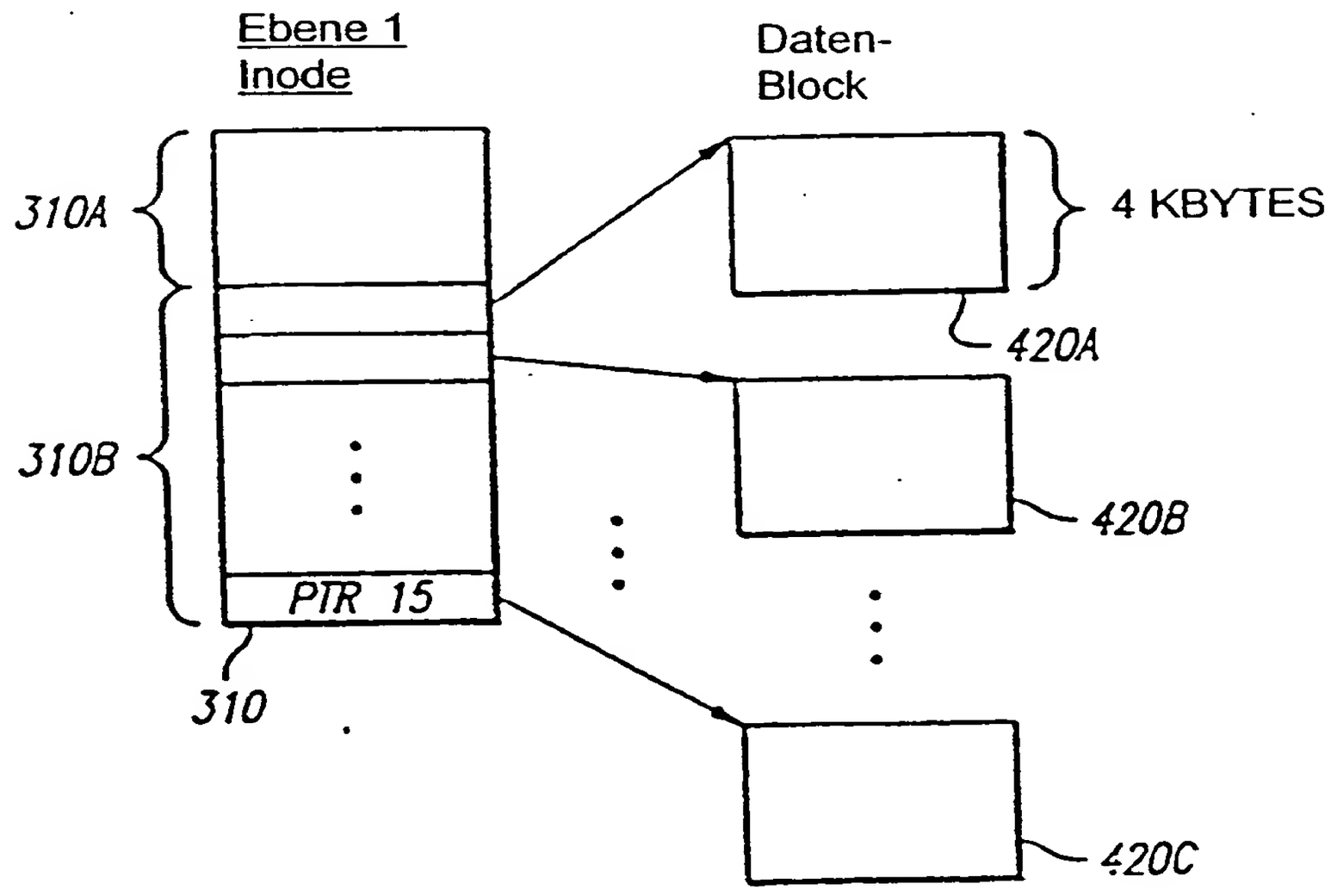

Ebene 0
Inode
4-Bytes Daten
4-Bytes Daten
4-Bytes Daten
410
510A
64 Bytes
Daten
410B

FIG. 4B


Ebene 1
Inode
Daten-
Block
310A
310B
PTR 15
310
4 KBYTES
420A
420B
420C

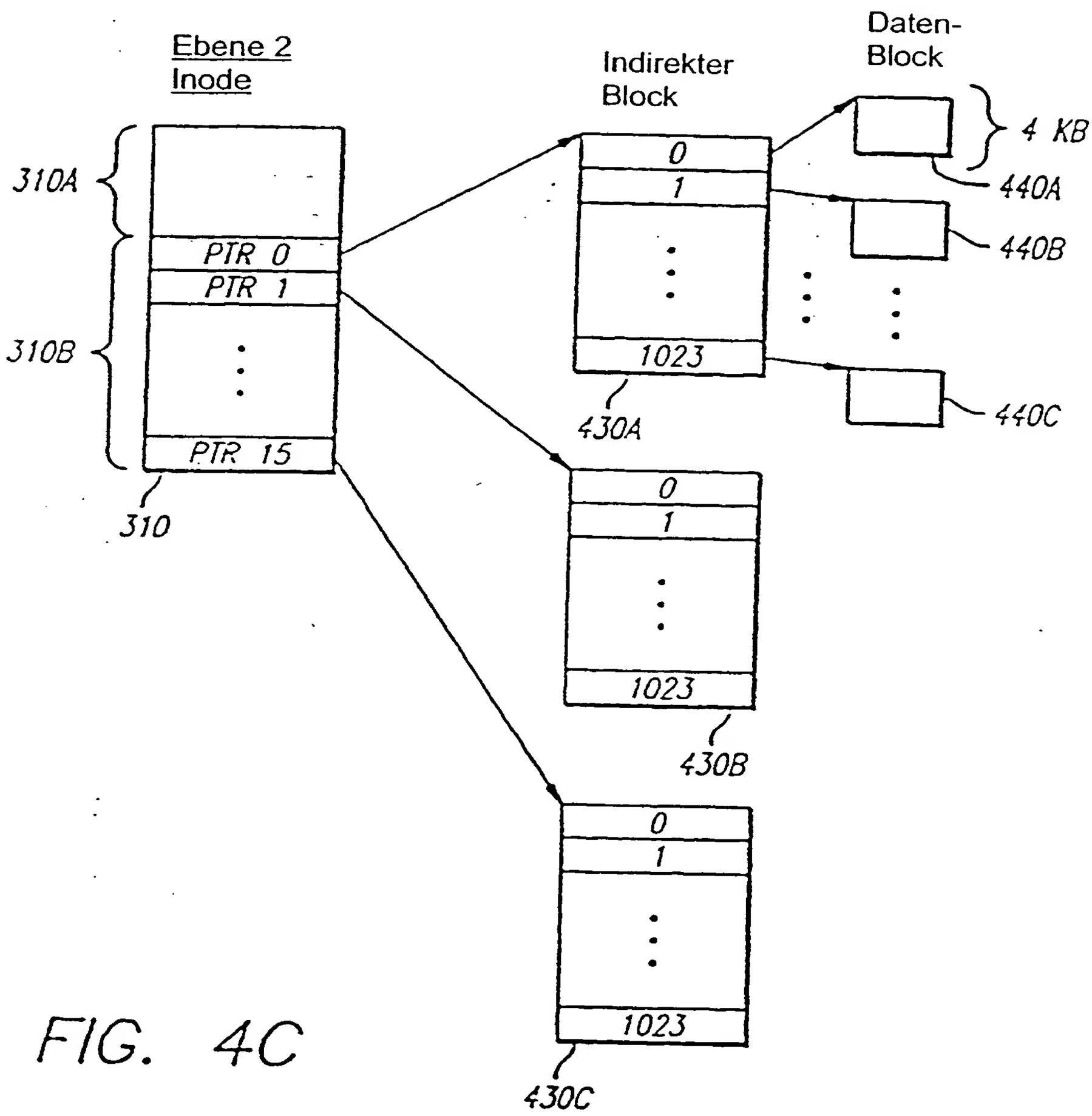

Ebene 2
Inode
310A
310B
PTR 0
PTR 1
PTR 15
310
Indirekter Block
0
1
1023
430A
Daten-Block
4 KB
440A
440B
440C
0
1
1023
430B
0
1
1023
430C

FIG. 4C

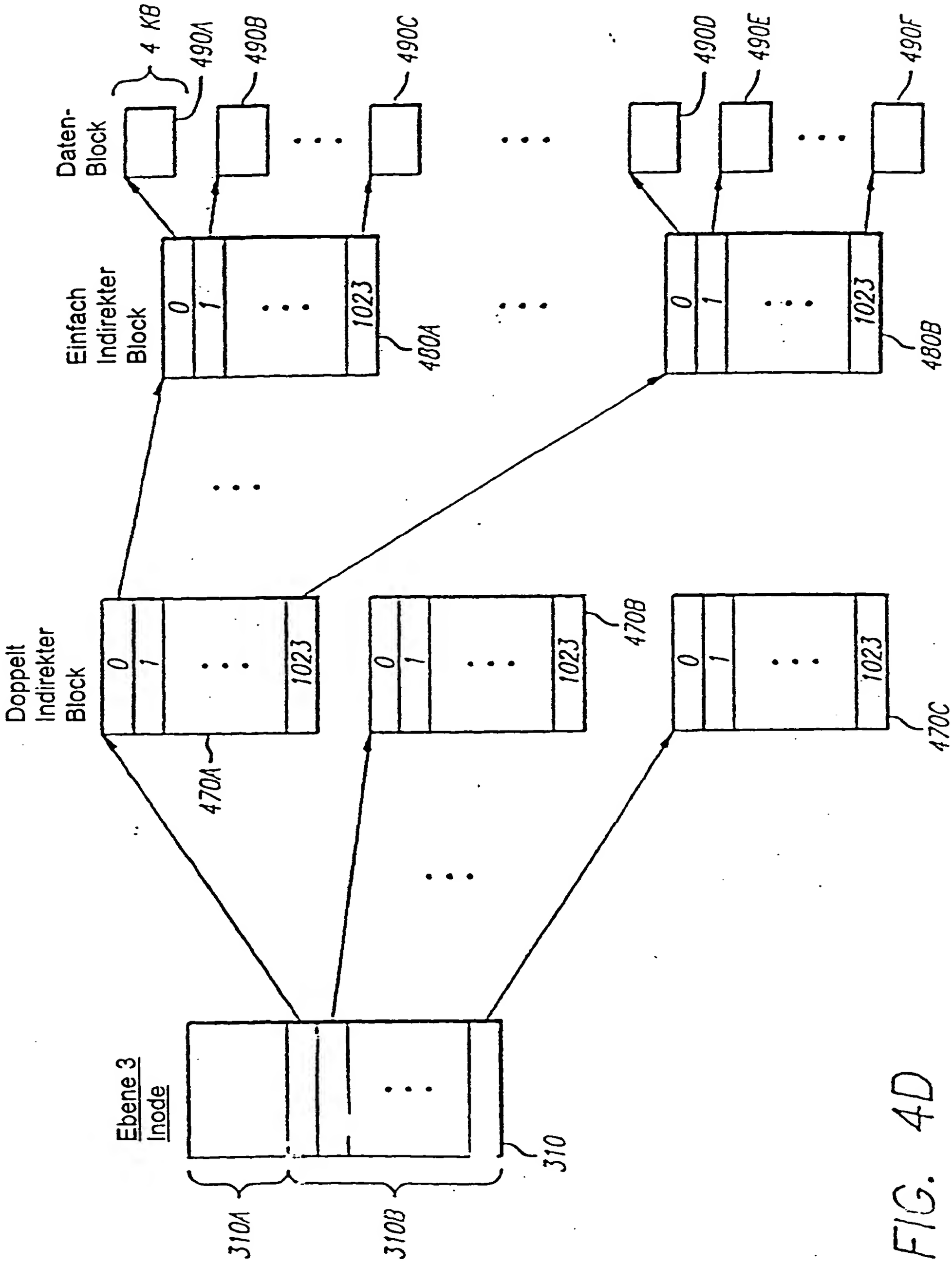

Ebene 3
Inode
310
310A
310B
Doppelt
Indirekter
Block
470A
470B
470C
Einfach
Indirekter
Block
480A
480B
Daten-
Block
4 KB
490A
490B
490C
490D
490E
490F
0
1
1023

FIG. 4D

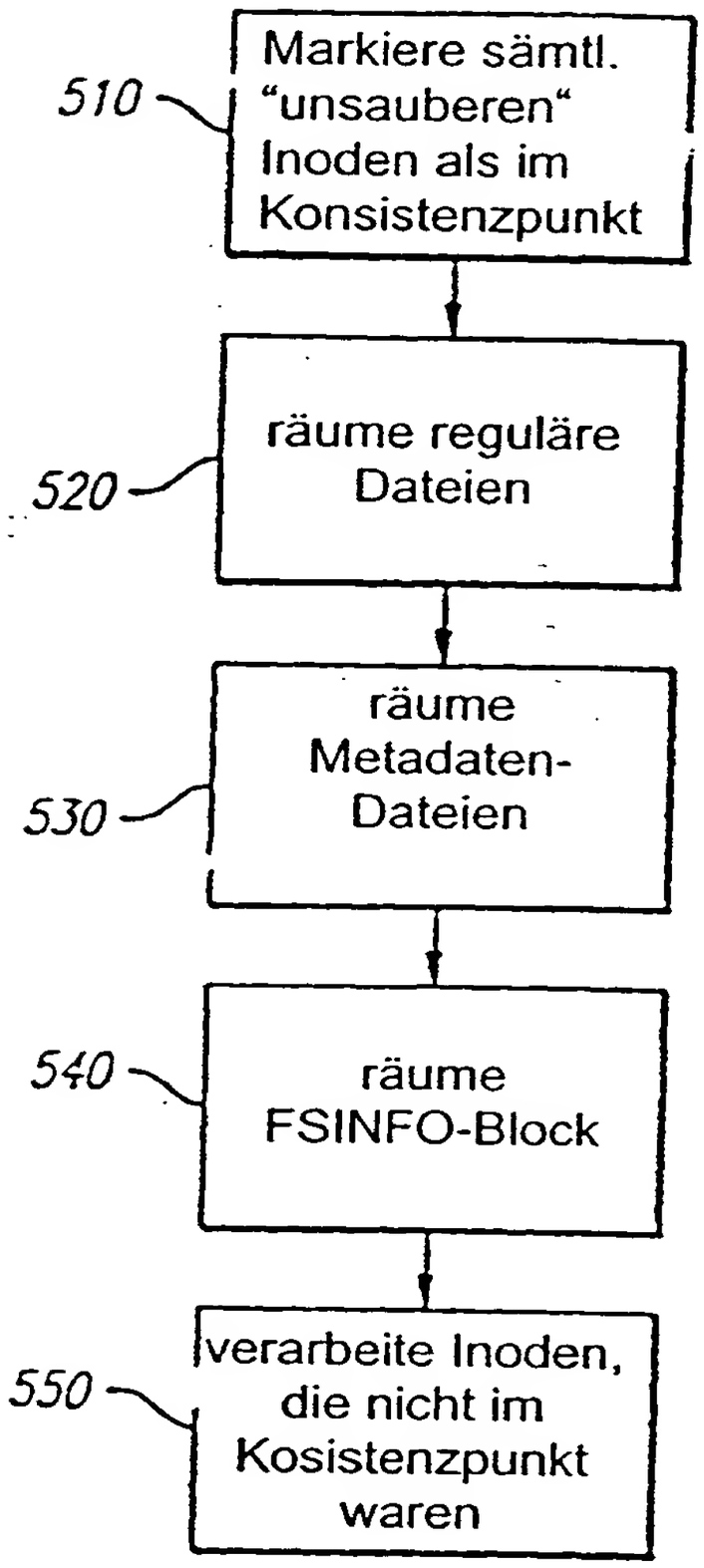

Markiere sämtl. "unsauberen" Inoden als im Konsistenzpunkt
510
räume reguläre Dateien
520
räume Metadaten-Dateien
530
räume FSINFO-Block
540
verarbeite Inoden, die nicht im Kosistenzpunkt waren
550

FIG. 5

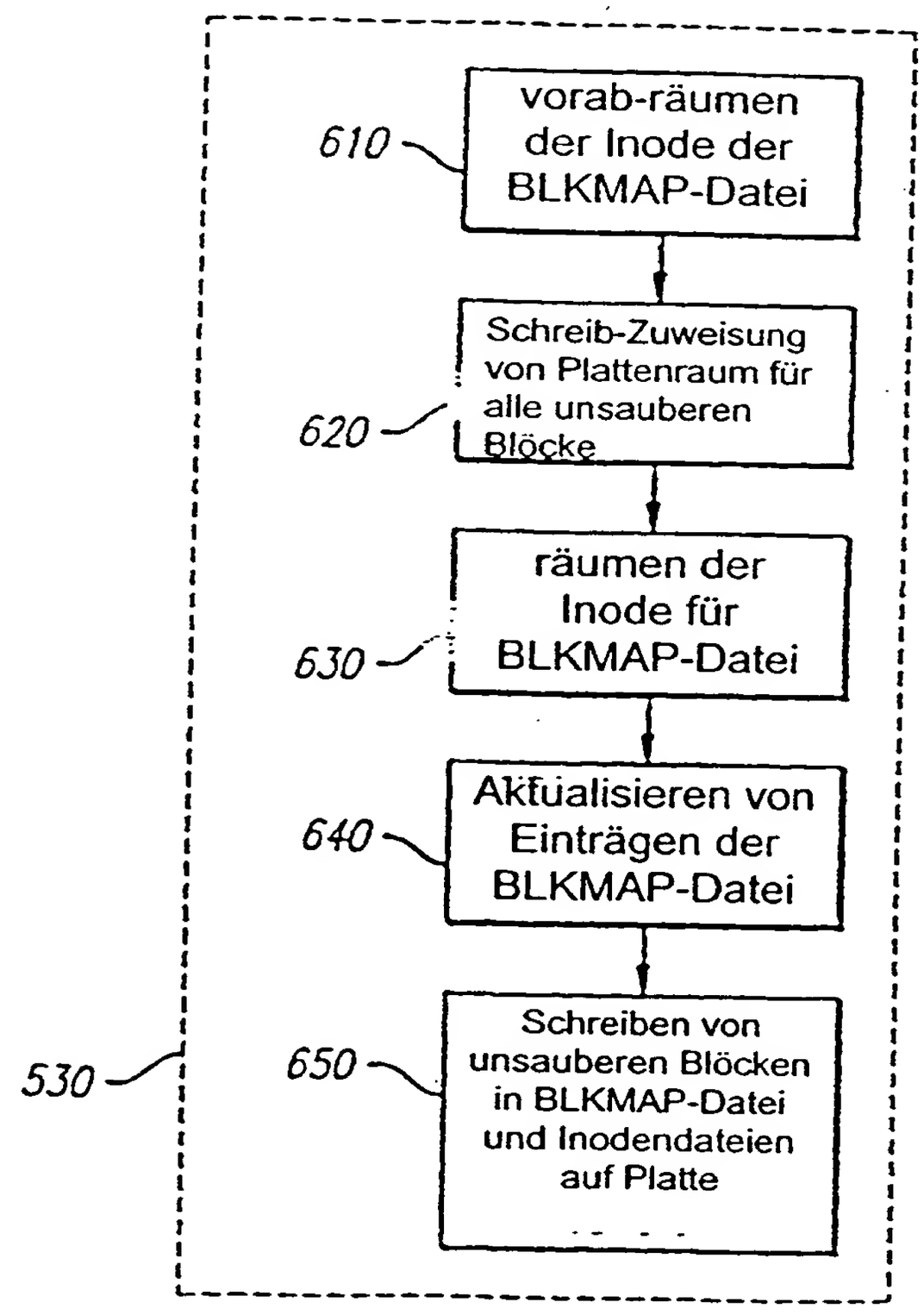

vorab-räumen der Inode der BLKMAP-Datei
610
Schreib-Zuweisung von Plattenraum für alle unsauberen Blöcke
620
räumen der Inode für BLKMAP-Datei
630
Aktualisieren von Einträgen der BLKMAP-Datei
640
Schreiben von unsauberen Blöcken in BLKMAP-Datei und Inodendateien auf Platte
650
530

FIG. 6

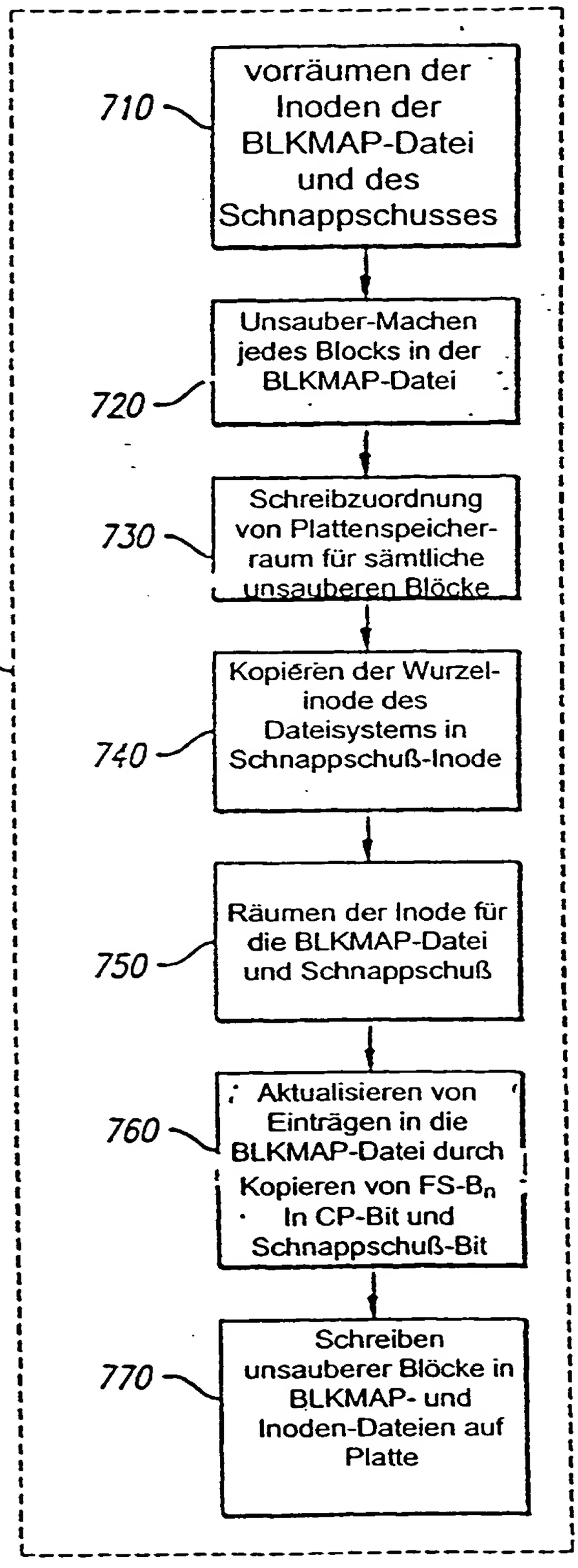

530
710
vorräumen der Inoden der BLKMAP-Datei und des Schnappschusses
720
Unsauber-Machen jedes Blocks in der BLKMAP-Datei
730
Schreibzuordnung von Plattenspeicher-raum für sämtliche unsauberen Blöcke
740
Kopieren der Wurzel-inode des Dateisystems in Schnappschuß-Inode
750
Räumen der Inode für die BLKMAP-Datei und Schnappschuß
760
Aktualisieren von Einträgen in die BLKMAP-Datei durch Kopieren von FS-B_n In CP-Bit und Schnappschuß-Bit
770
Schreiben unsauberer Blöcke in BLKMAP- und Inoden-Dateien auf Platte

FIG. 7

FIG. 8

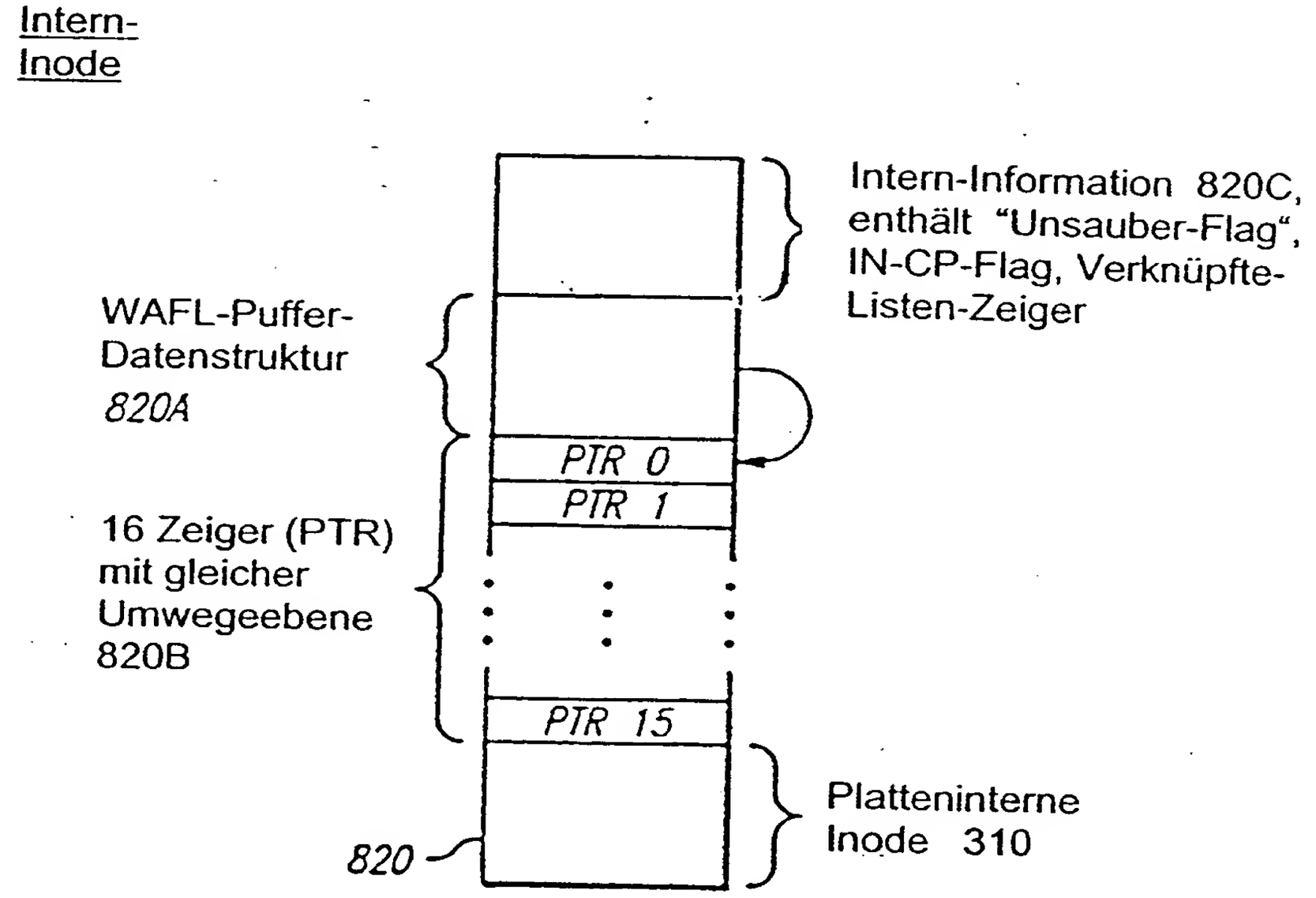

Intern-
Inode
Intern-Information 820C, enthält "Unsauber-Flag", IN-CP-Flag, Verknüpfte-Listen-Zeiger
WAFL-Puffer-Datenstruktur
820A
PTR 0
PTR 1
16 Zeiger (PTR) mit gleicher Umwegeebene 820B
PTR 15
Platteninterne Inode 310
820

FIG. 9A

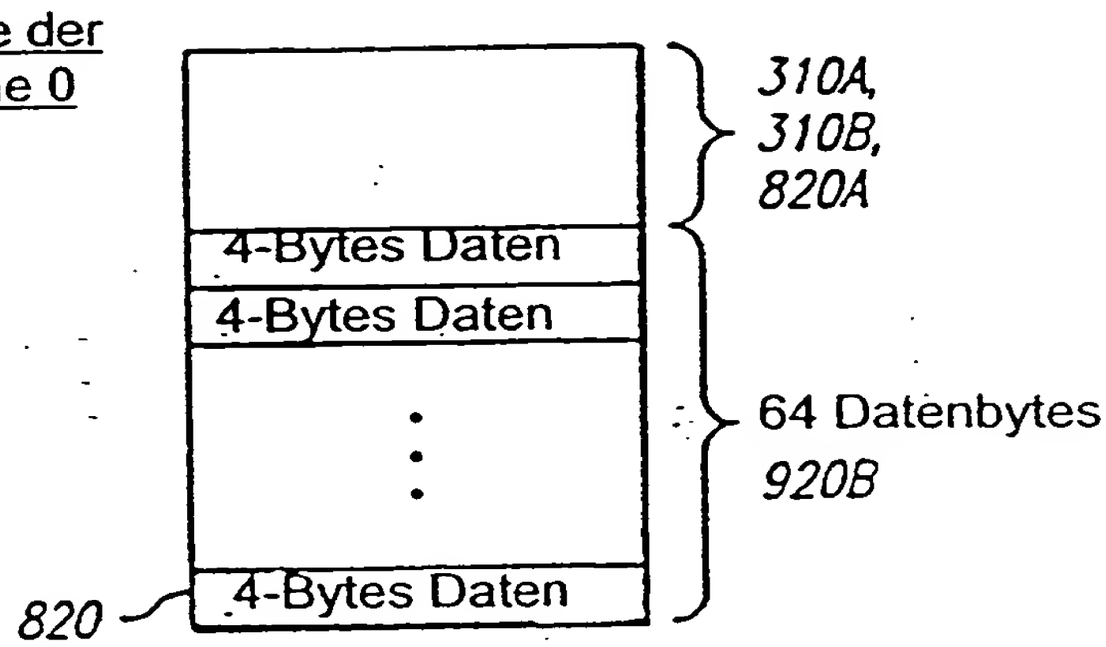
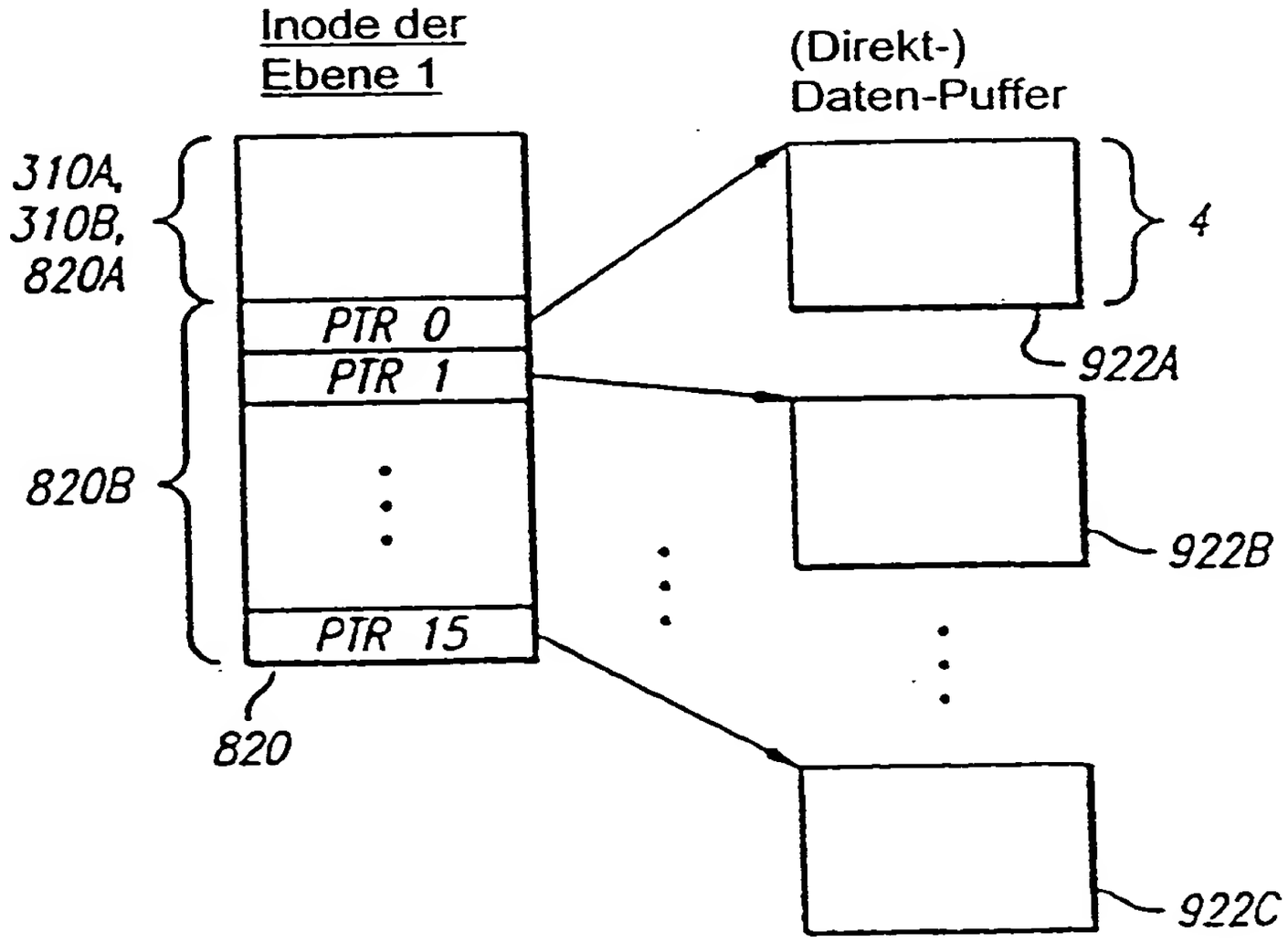

Inode der
Ebene 0
4-Bytes Daten
4-Bytes Daten
4-Bytes Daten
820
310A,
310B,
820A
64 Datenbytes
920B

FIG. 9B


Inode der
Ebene 1
(Direkt-)
Daten-Puffer
310A,
310B,
820A
820B
PTR 0
PTR 1
PTR 15
820
4
922A
922B
922C

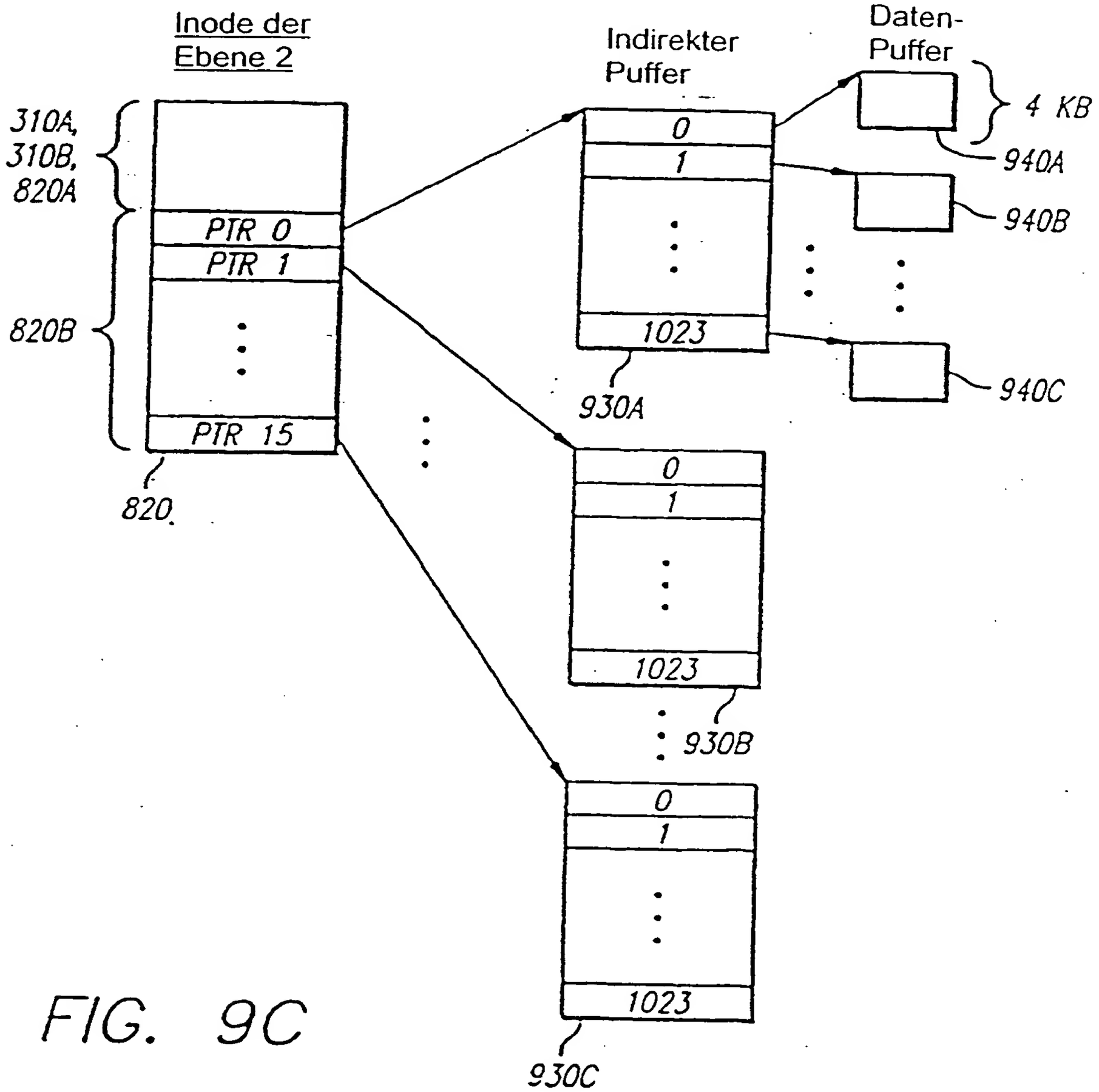

Inode der Ebene 2
Indirekter Puffer
Daten-Puffer
310A, 310B, 820A
820B
PTR 0
PTR 1
PTR 15
820
0
1
1023
930A
930B
930C
4 KB
940A
940B
940C

FIG. 9C

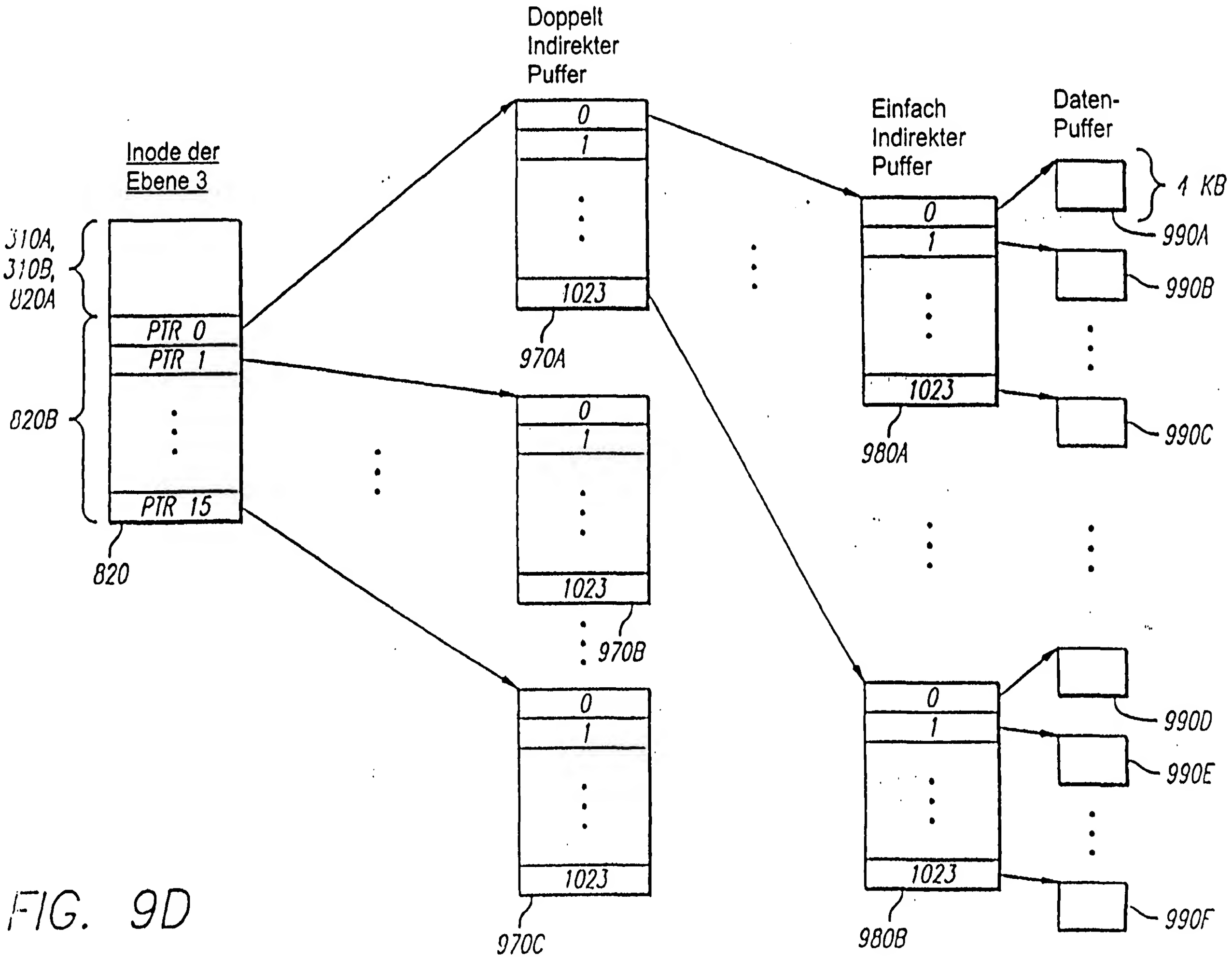

Inode der
Ebene 3
310A,
310B,
820A
820B
PTR 0
PTR 1
PTR 15
820
Doppelt
Indirekter
Puffer
0
1
1023
970A
970B
970C
Einfach
Indirekter
Puffer
980A
980B
Daten-
Puffer
1 KB
990A
990B
990C
990D
990E
990F

FIG. 9D

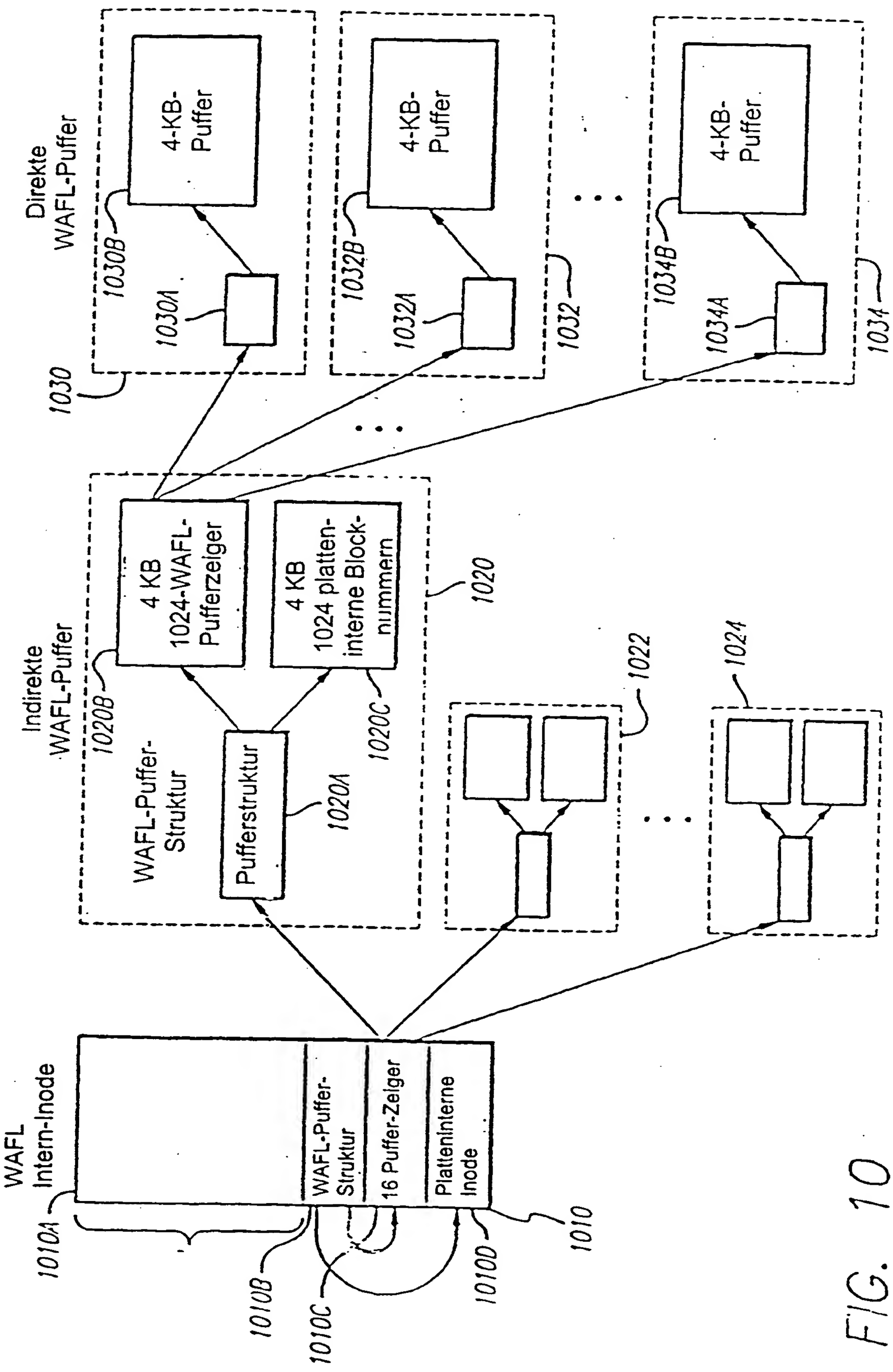

WAFL
Intern-Inode
1010A
1010B
1010C
1010D
1010
WAFL-Puffer-
Struktur
16 Puffer-Zeiger
Plattenlnterne
Inode
Indirekte
WAFL-Puffer
1020B
4 KB
1024-WAFL-
Pufferzeiger
4 KB
1024 platten-
interne Block-
nummern
1020C
WAFL-Puffer-
Struktur
Pufferstruktur
1020A
1020
1022
1024
Direkte
WAFL-Puffer
1030B
4-KB-
Puffer
1030A
1030
1032B
4-KB-
Puffer
1032A
1032
1034B
4-KB-
Puffer
1034A
1034

FIG. 10

FIG. 11A

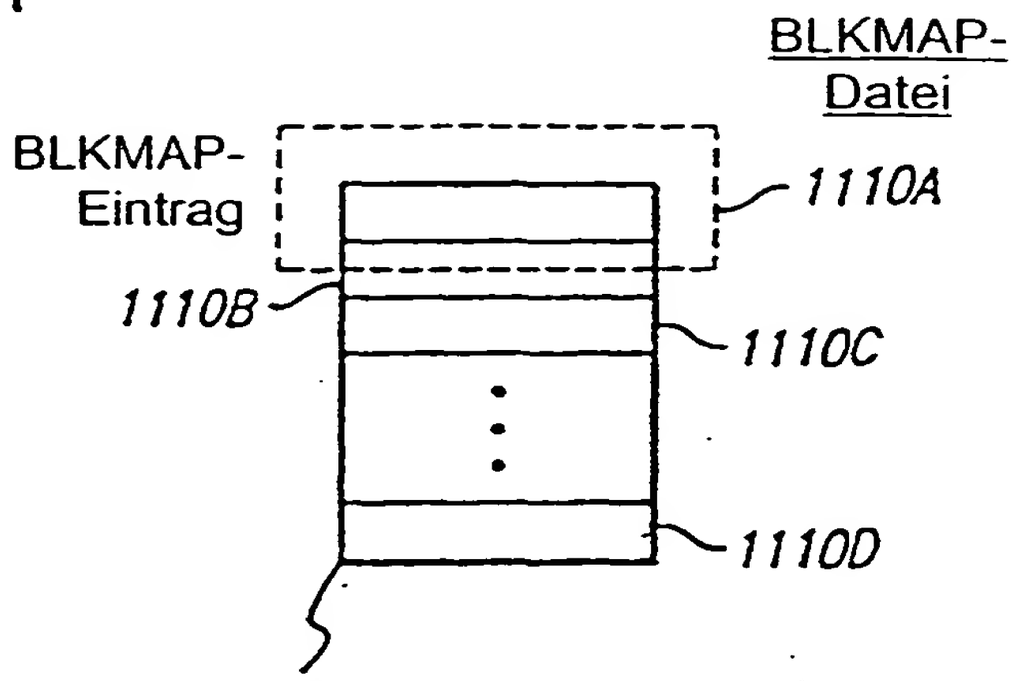
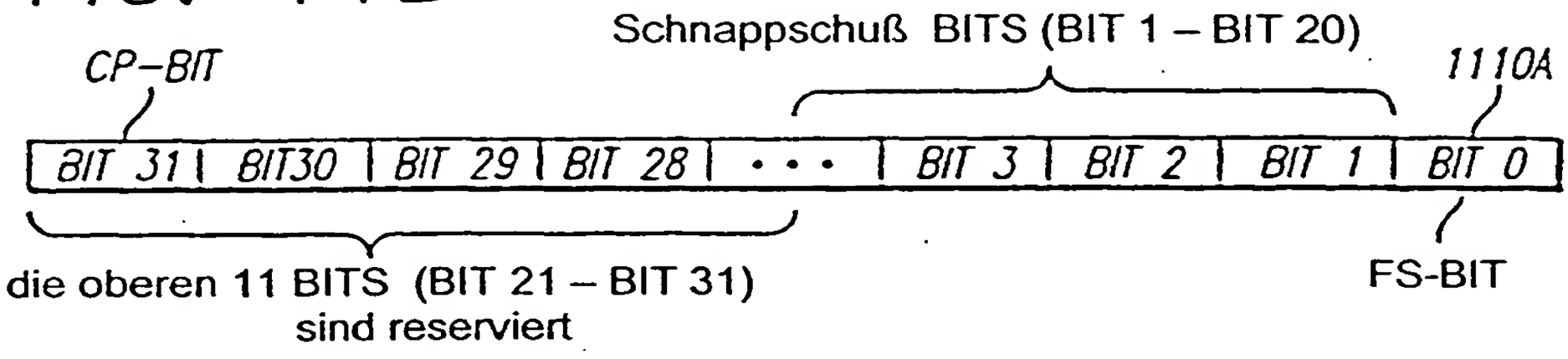


FIG. 11B



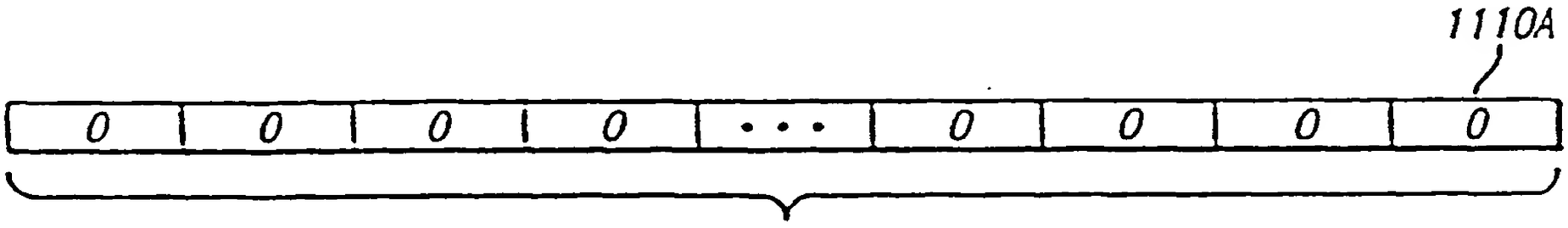


FIG. 11C

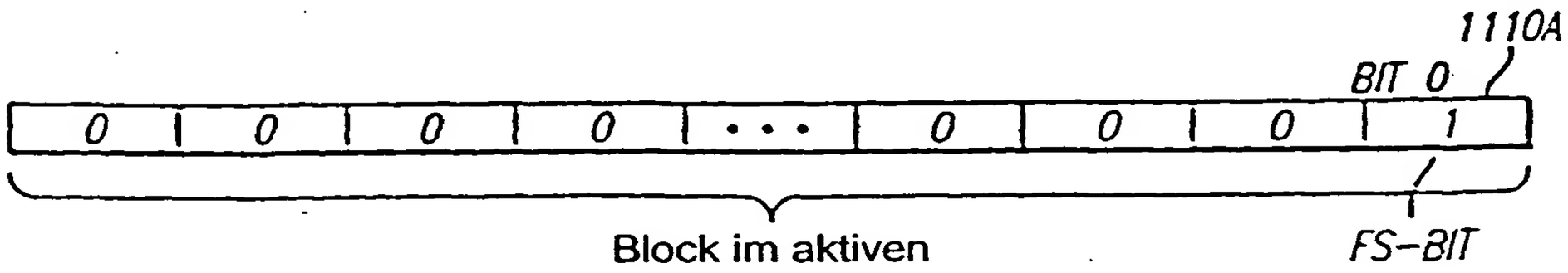


FIG. 11D

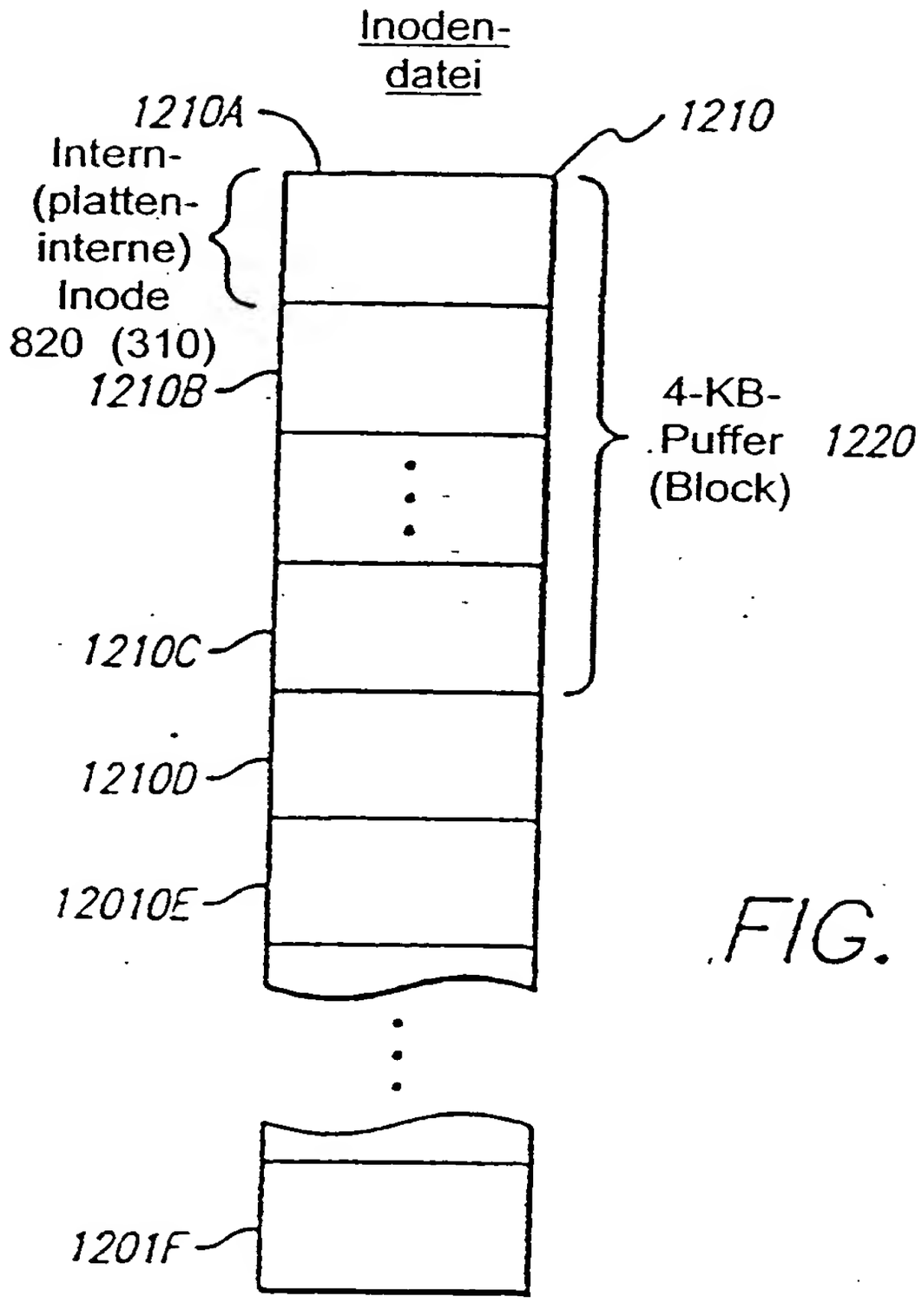

Inoden-
datei
1210A
1210
Intern-
(platten-
interne)
Inode
820 (310)
1210B
4-KB-
Puffer
(Block)
1220
1210C
1210D
12010E
1201F

FIG. 12

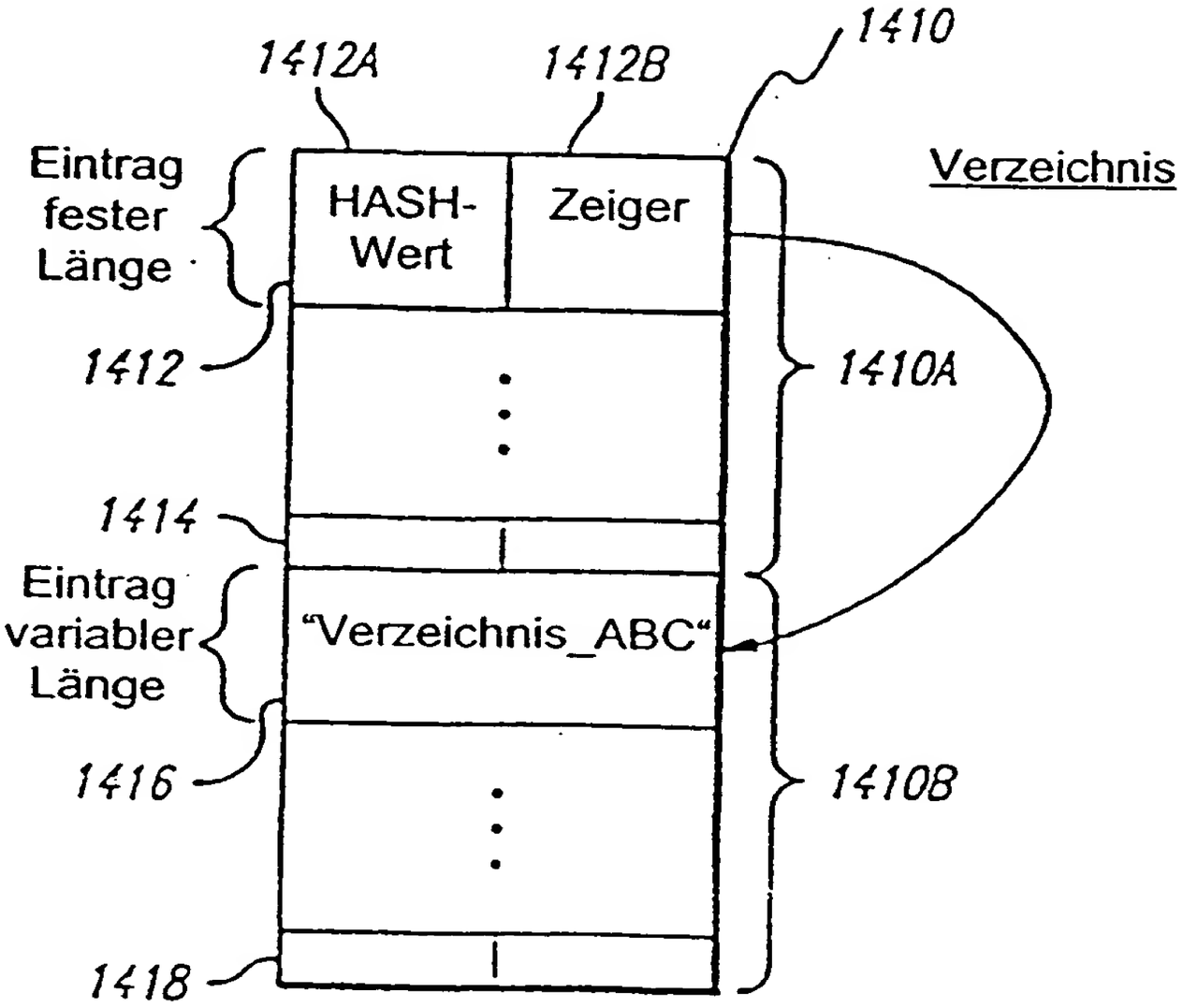
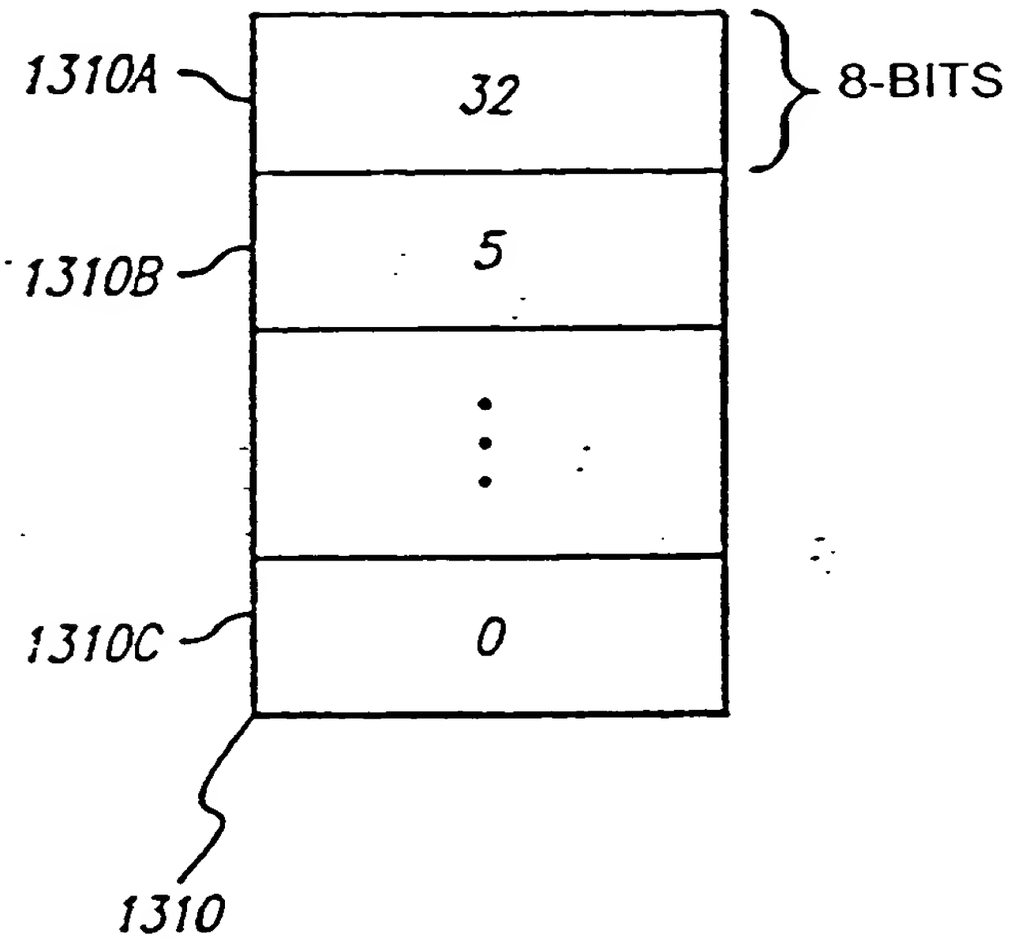

1412A
1412B
1410
Verzeichnis
Eintrag
fester
Länge
HASH-
Wert
Zeiger
1412
1410A
1414
Eintrag
variabler
Länge
"Verzeichnis_ABC"
1416
14108
1418

FIG. 14

FIG. 13A



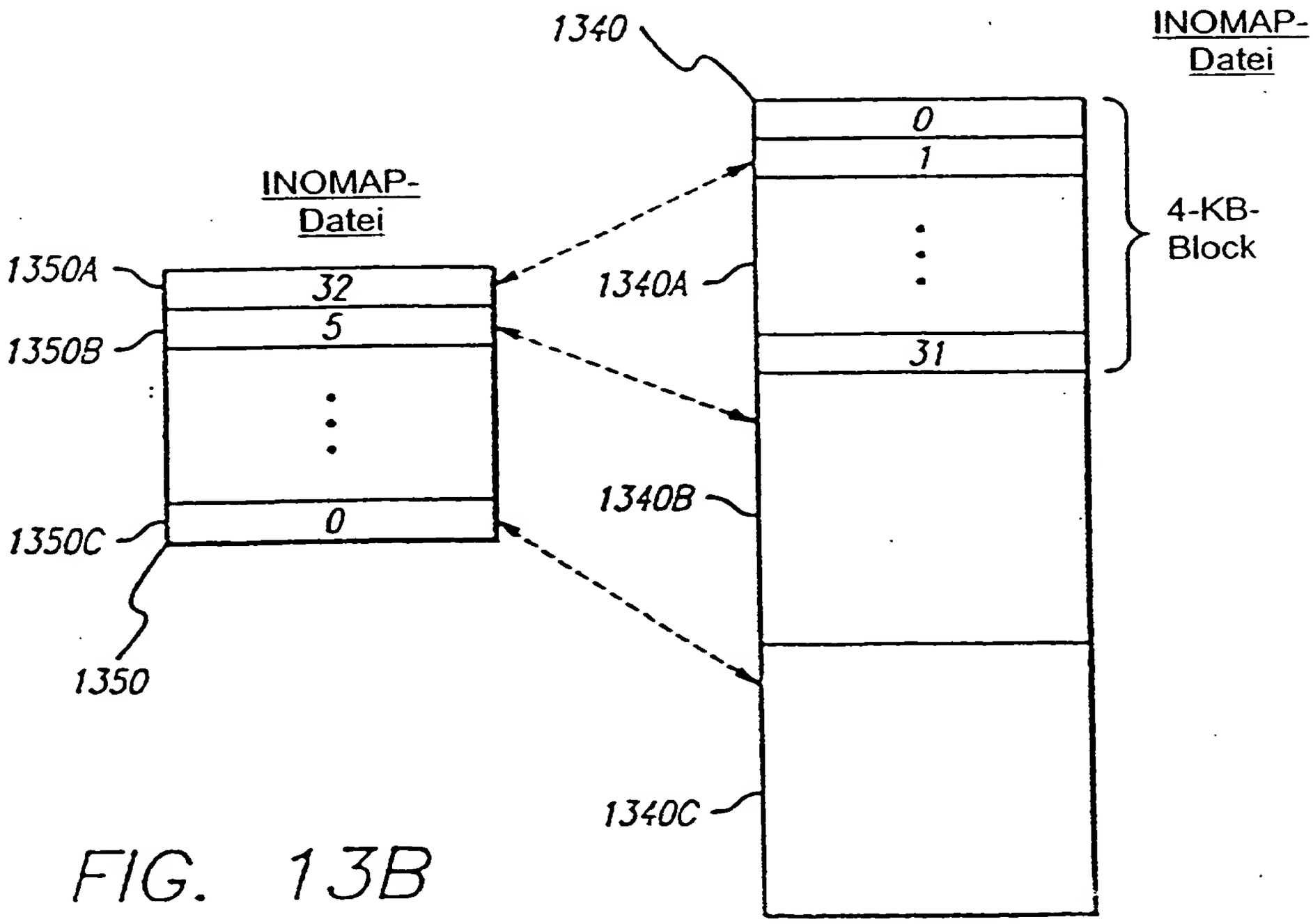


FIG. 13B

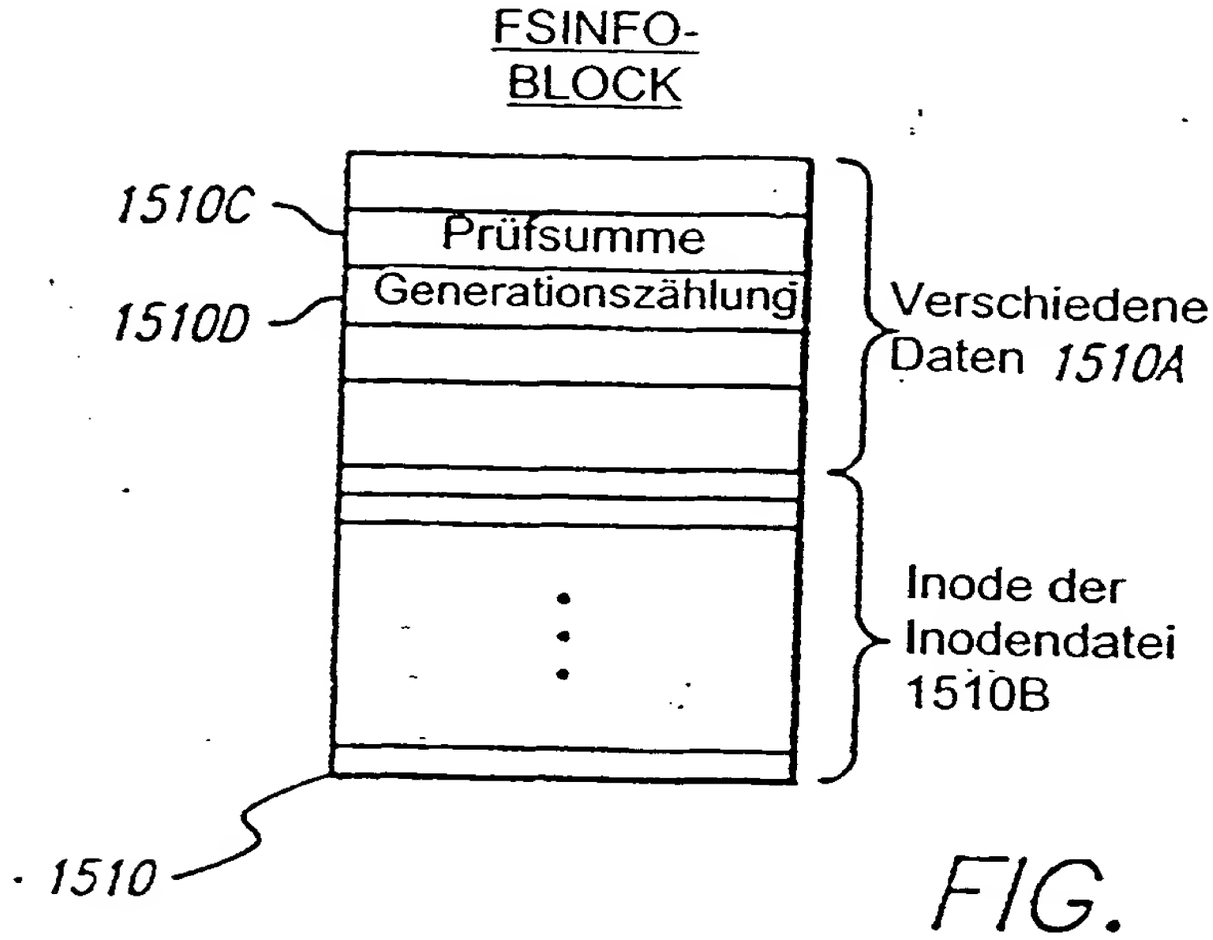


FIG. 15

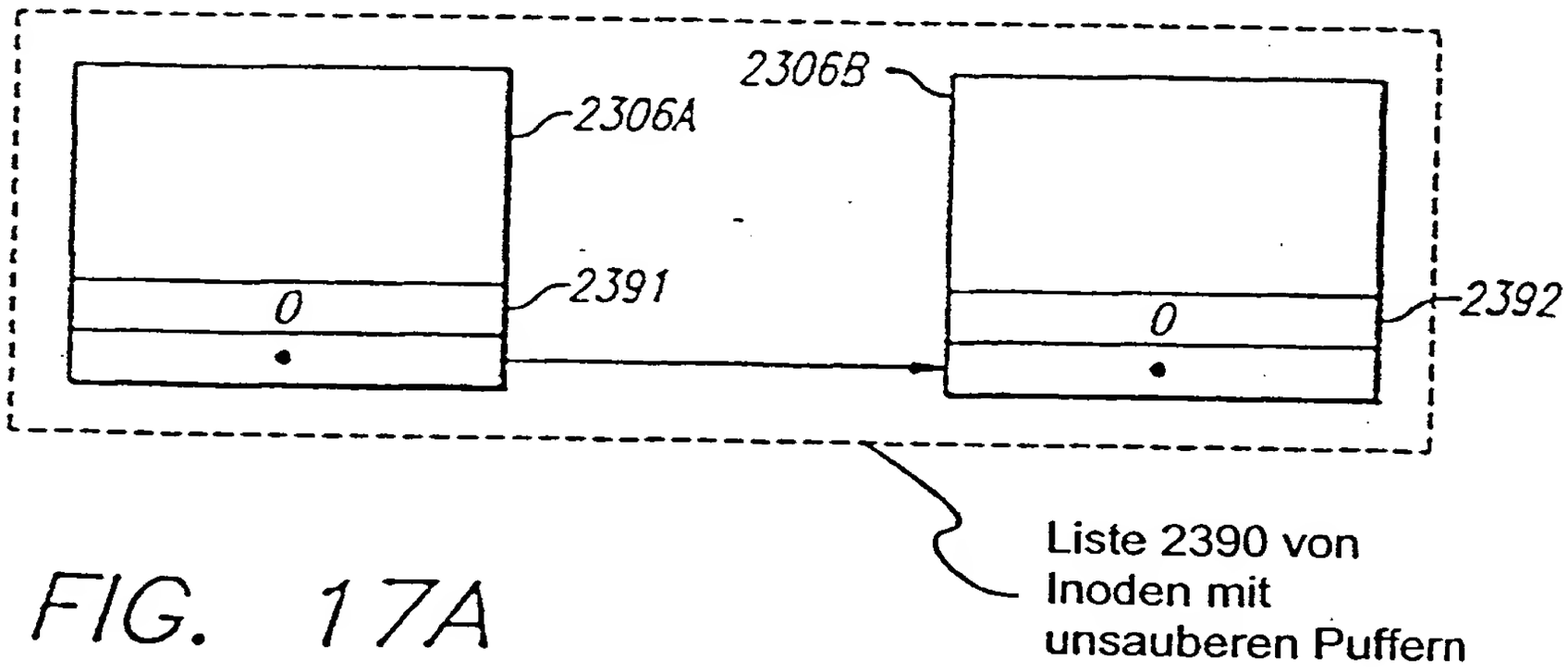


FIG. 17A

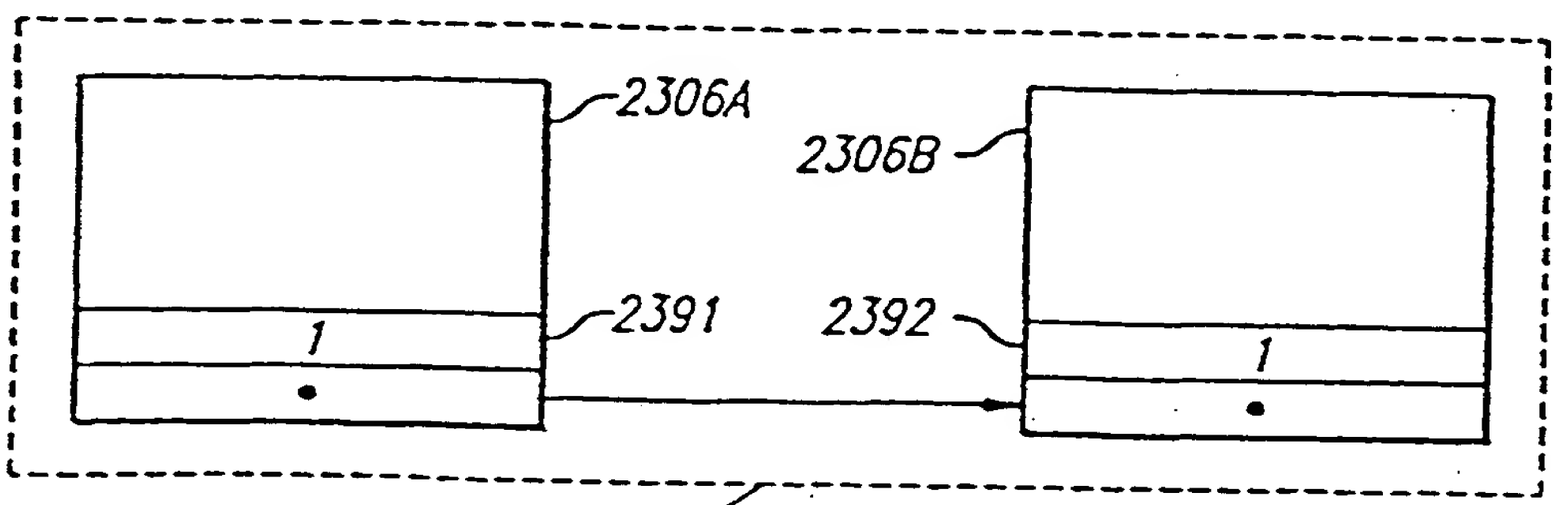


FIG. 17I

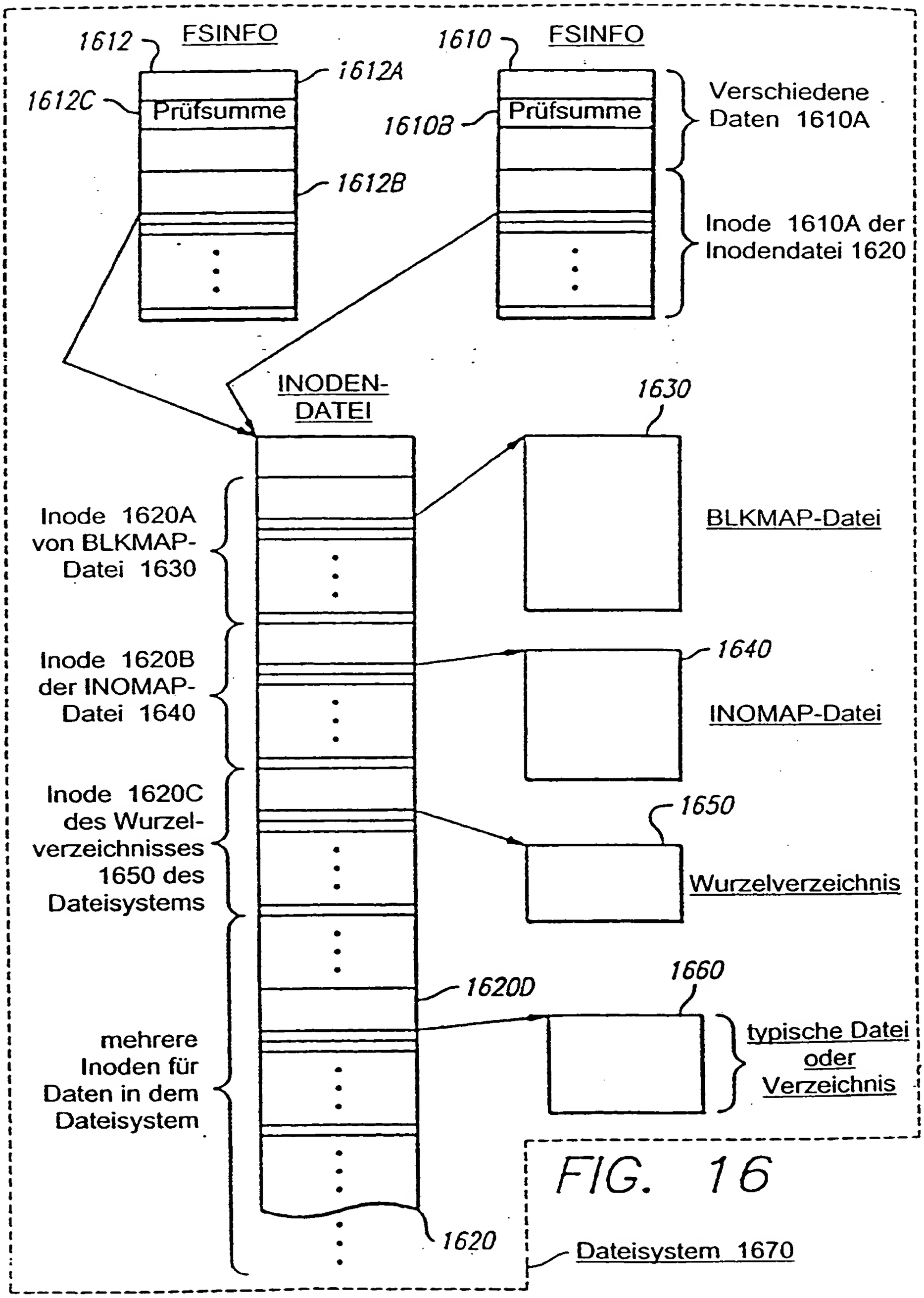

1612
FSINFO
1612A
1612C
Prüfsumme
1612B
1610
FSINFO
Prüfsumme
1610B
Verschiedene Daten 1610A
Inode 1610A der Inodendatei 1620
INODEN-DATEI
1630
Inode 1620A von BLKMAP-Datei 1630
BLKMAP-Datei
1640
Inode 1620B der INOMAP-Datei 1640
INOMAP-Datei
1650
Inode 1620C des Wurzel-verzeichnisses 1650 des Dateisystems
Wurzelverzeichnis
1660
16200
mehrere Inoden für Daten in dem Dateisystem
typische Datei oder Verzeichnis
1620
Dateisystem 1670

FIG. 16

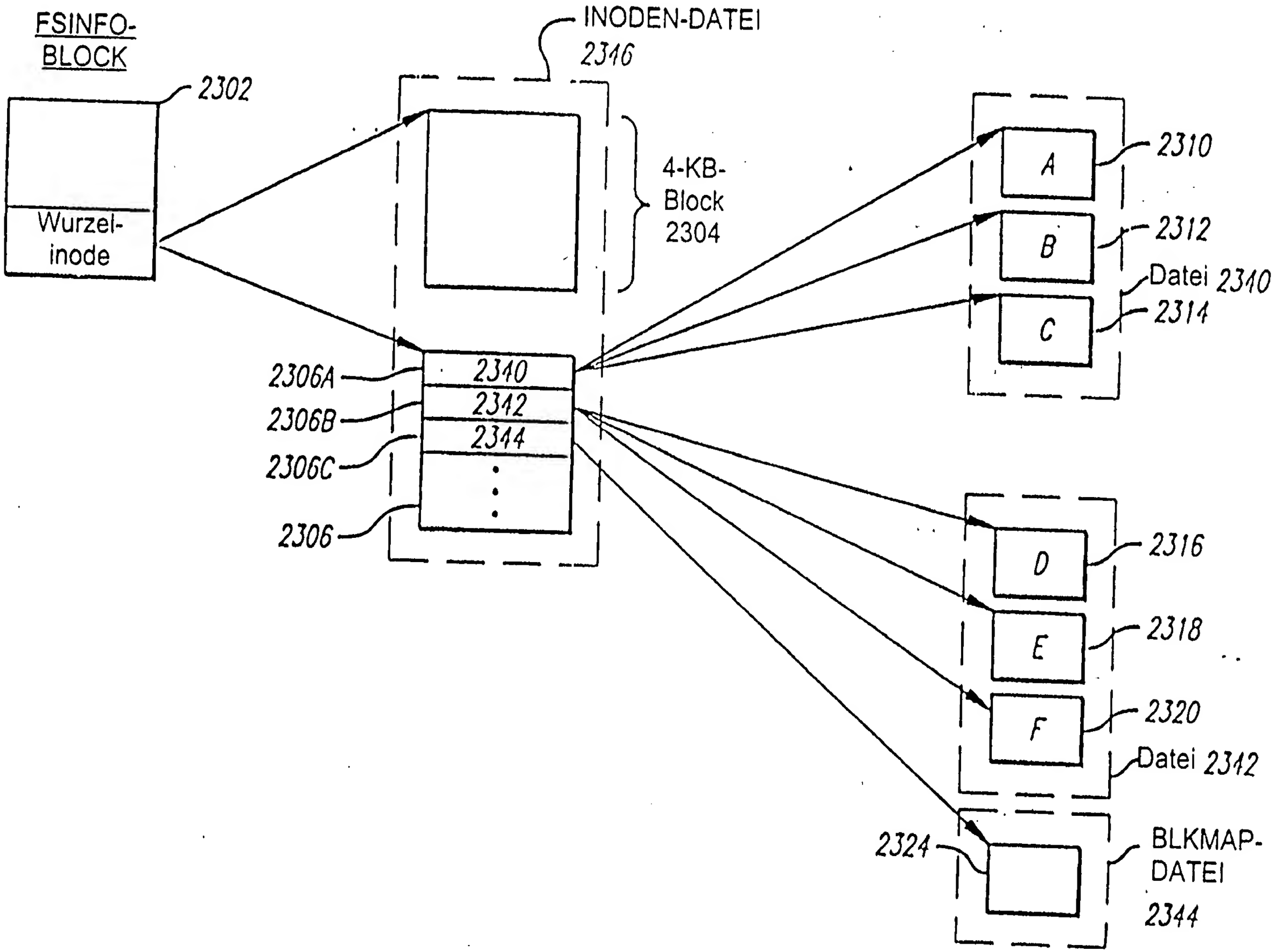

FSINFO-
BLOCK
2302
Wurzel-
inode
INODEN-DATEI
2316
4-KB-
Block
2304
2306A
2306B
2306C
2306
2310
2312
2314
2340
2342
2344
A
B
C
Datei 2340
2316
D
E
F
2318
2320
Datei 2342
2324
BLKMAP-
DATEI
2344

FIG. 17B

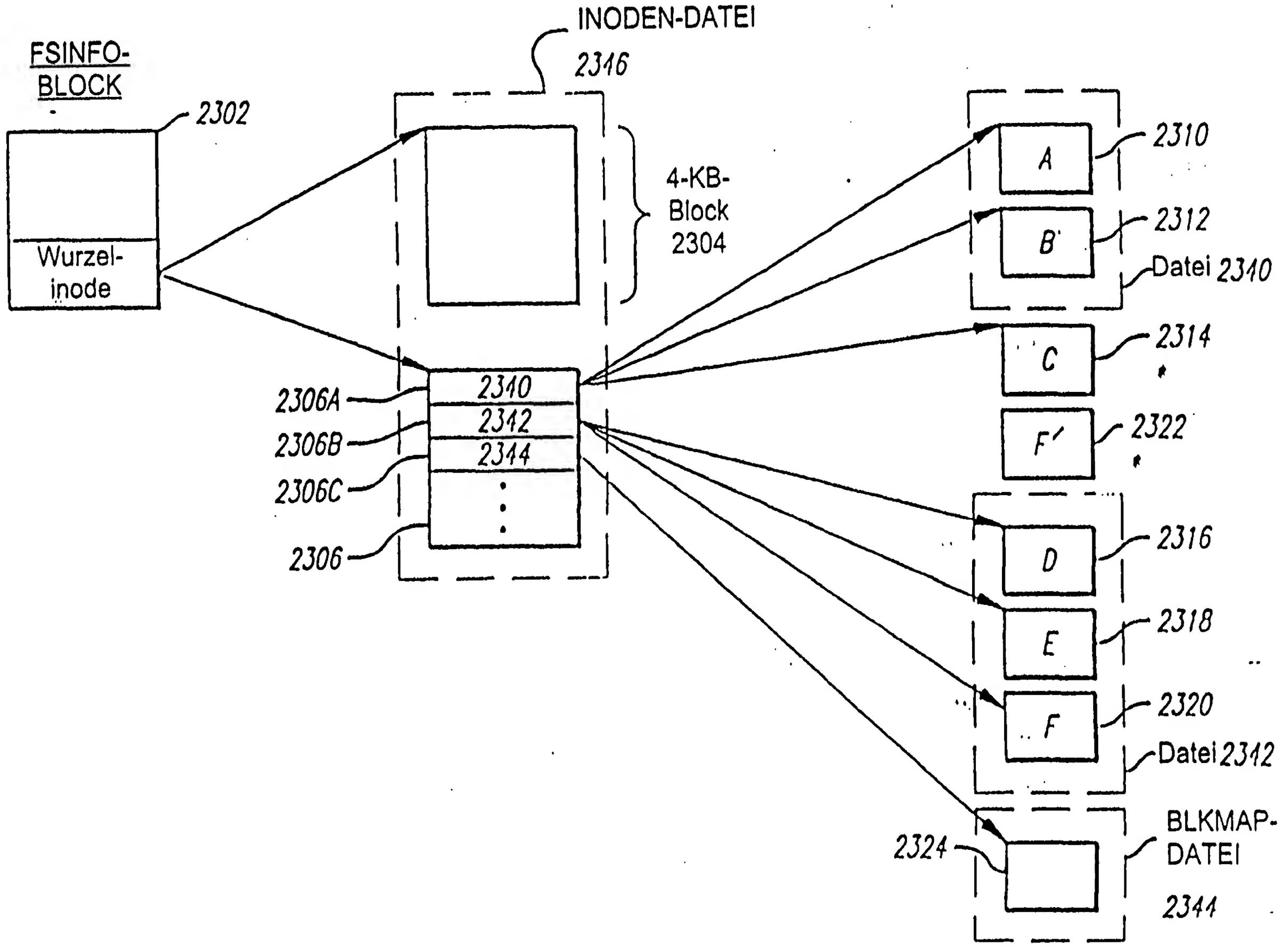

FSINFO-
BLOCK
2302
Wurzel-
inode
INODEN-DATEI
2316
4-KB-
Block
2304
2306A
2306B
2306C
2306
2310
2312
2314
2322
2316
2318
2320
2324
2341
A
B
C
F'
D
E
F
Datei 2310
Datei 2312
BLKMAP-
DATEI

FIG. 17C

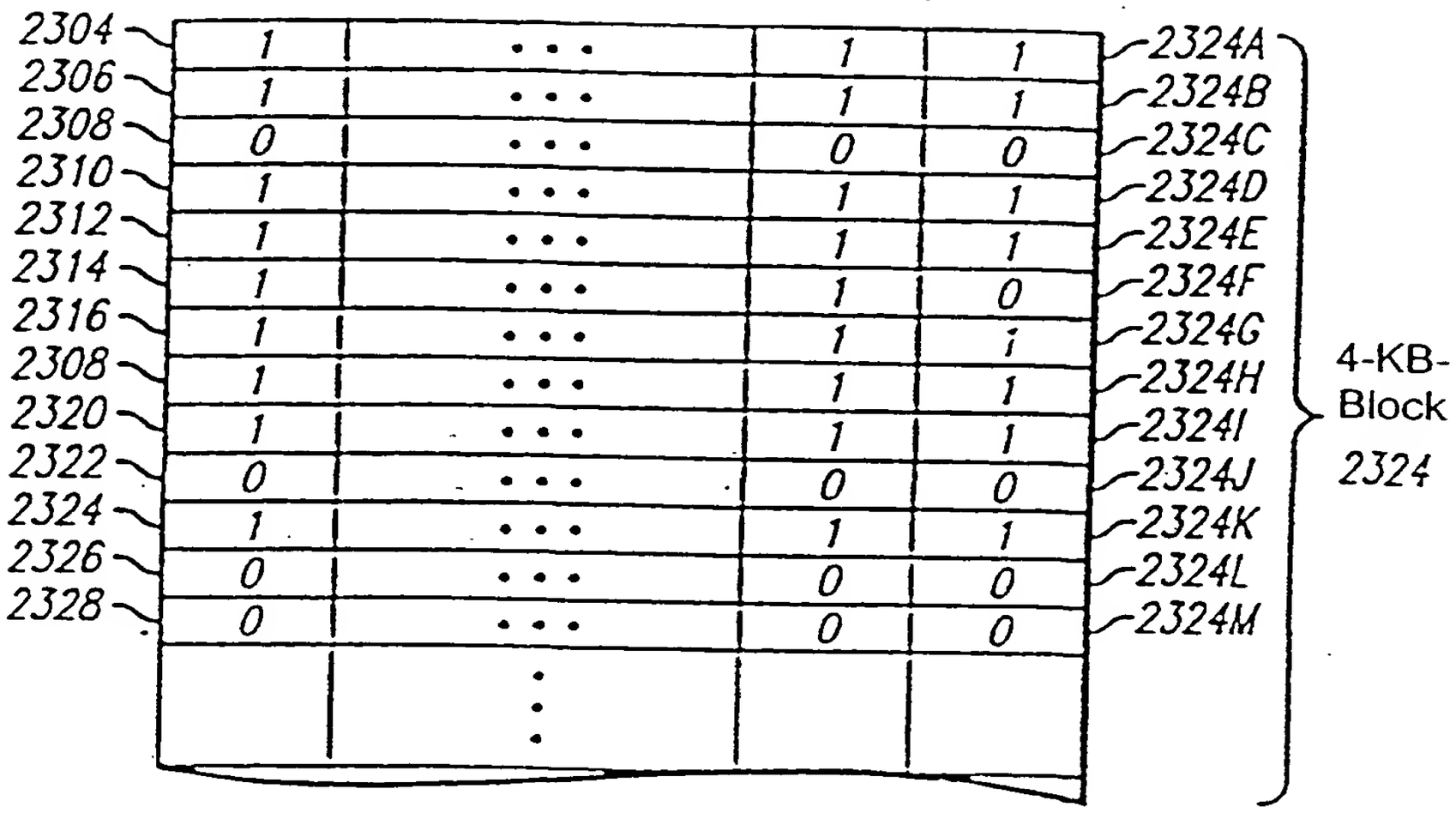


FIG. 17D

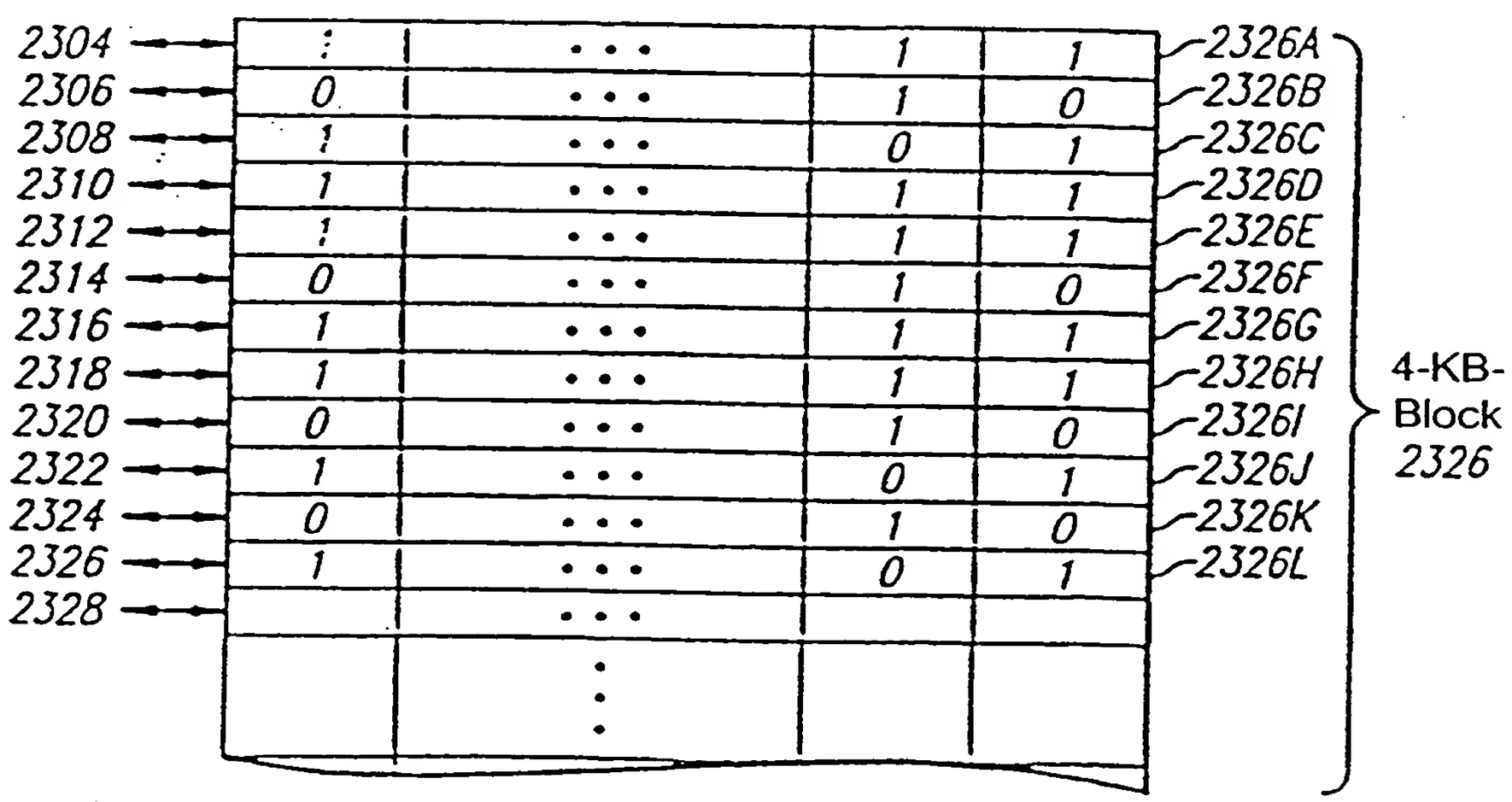


FIG. 17J

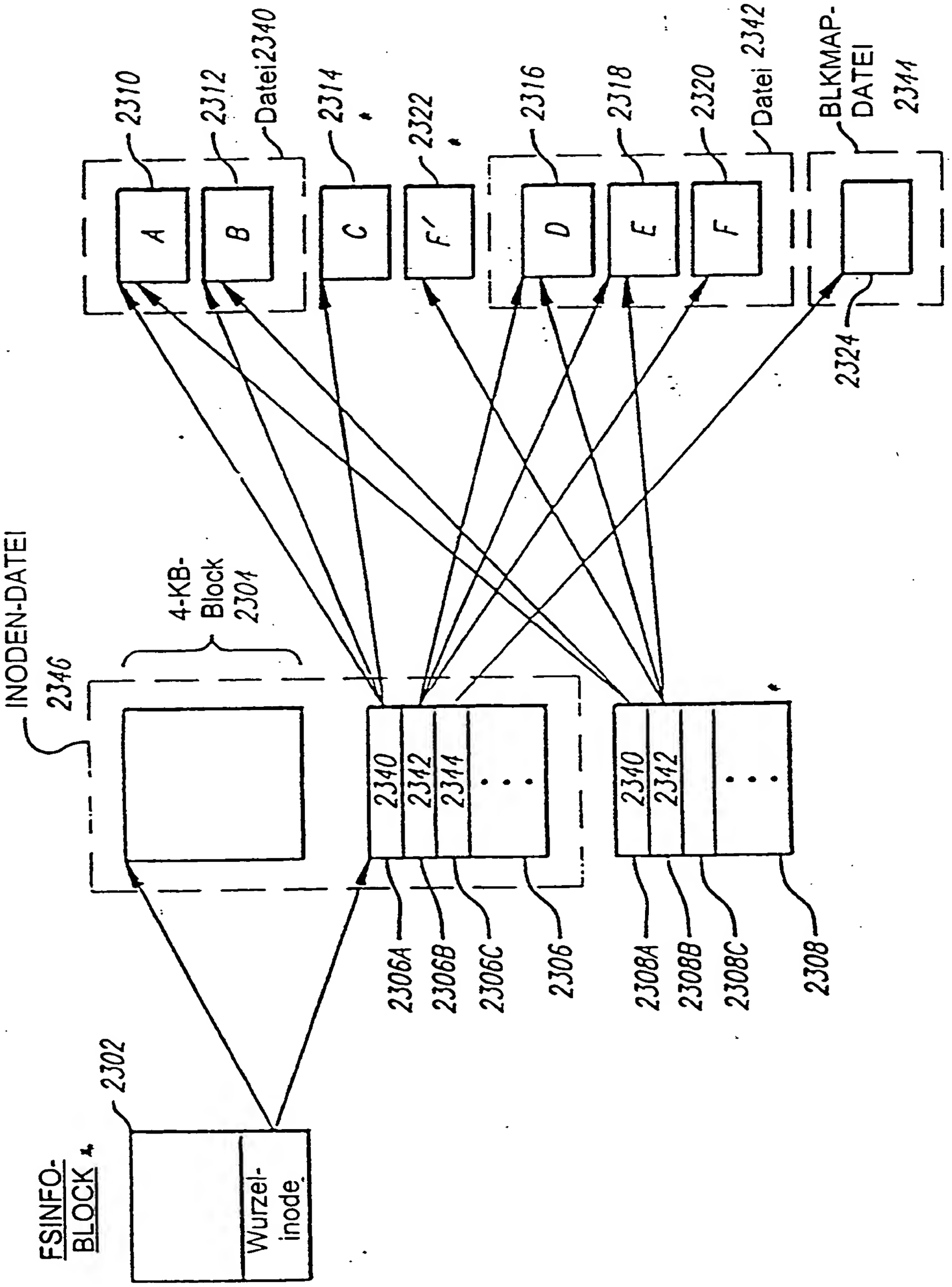

FSINFO-BLOCK
2302
Wurzel-inode
INODEN-DATEI
2346
4-KB-Block
2304
2310
2312
Datei2340
2314
2322
2316
2318
2320
Datei 2342
BLKMAP-DATEI
2344
A
B
C
F'
D
E
F
2324
2340
2342
2344
2306A
2306B
2306C
2306
2308A
2308B
2308C
2308

FIG. 17E

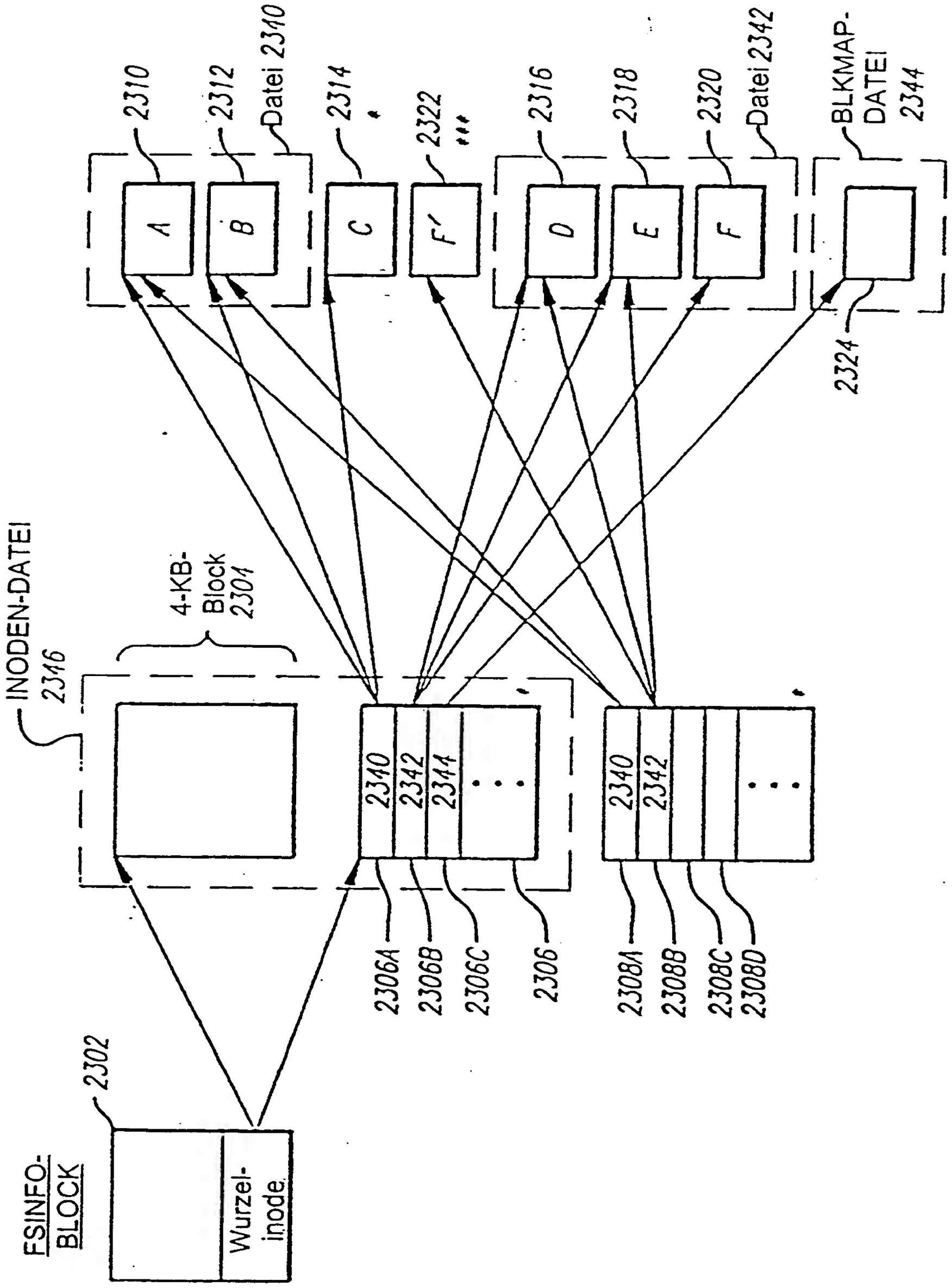

2310
2312
Datei 2310
2314
2322
2316
2318
2320
Datei 2342
BLKMAP-DATEI 2344
A
B
C
F'
D
E
F
2324
INODEN-DATEI 2346
4-KB-Block 2304
2340
2342
2344
2340
2342
2306A
2306B
2306C
2306
2308A
2308B
2308C
2308D
FSINFO-BLOCK
2302
Wurzel-inode

FIG. 17F

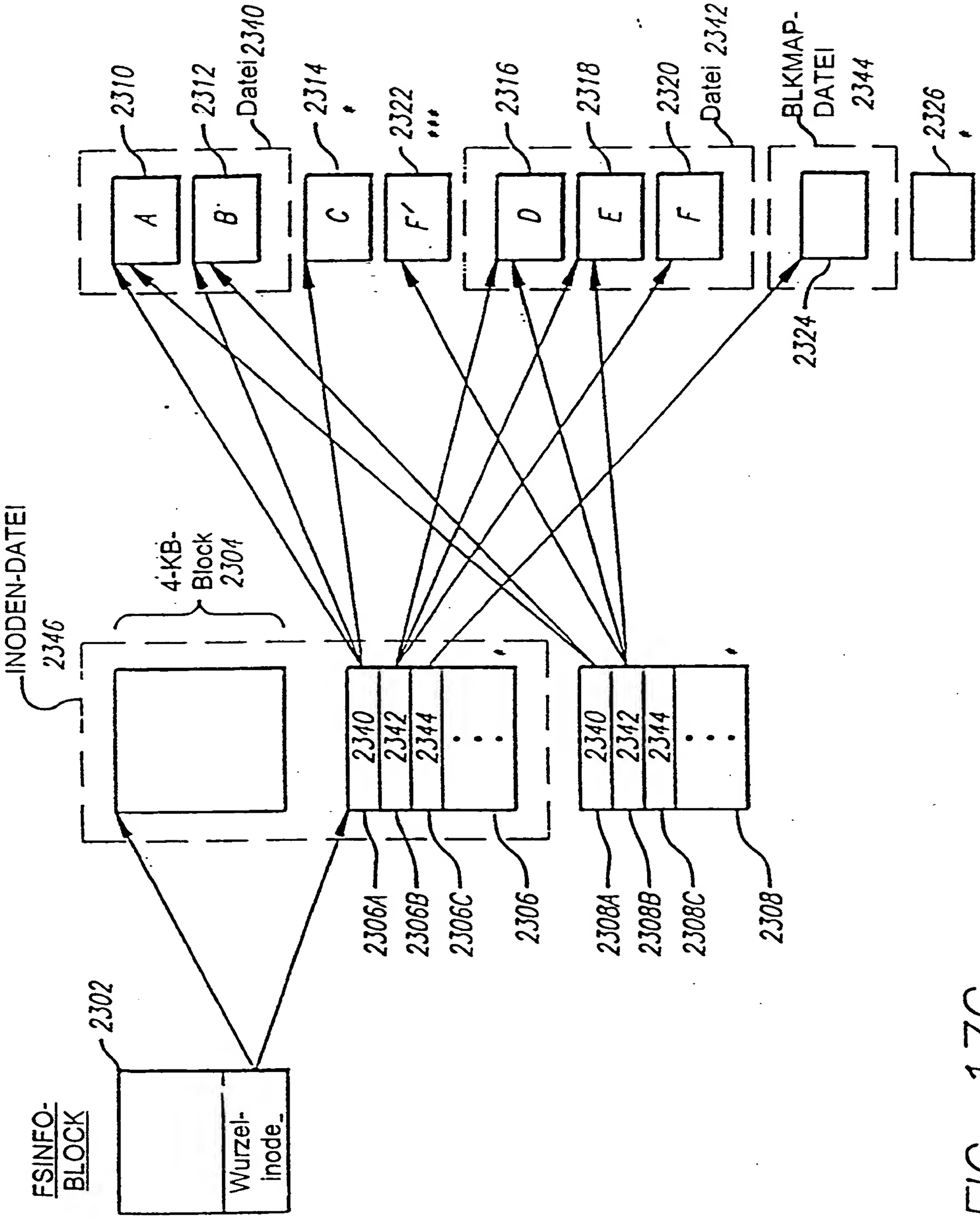

FSINFO-BLOCK
2302
Wurzel-inode_
INODEN-DATEI
2346
4-KB-Block 2304
2310
2312
Datei 2310
2314
2322
2316
2318
2320
Datei 2342
BLKMAP-DATEI 2344
2326
2324
A
B'
C
F'
D
E
F
2340
2342
2344
2306A
2306B
2306C
2306
2308A
2308B
2308C
2308

FIG. 17G

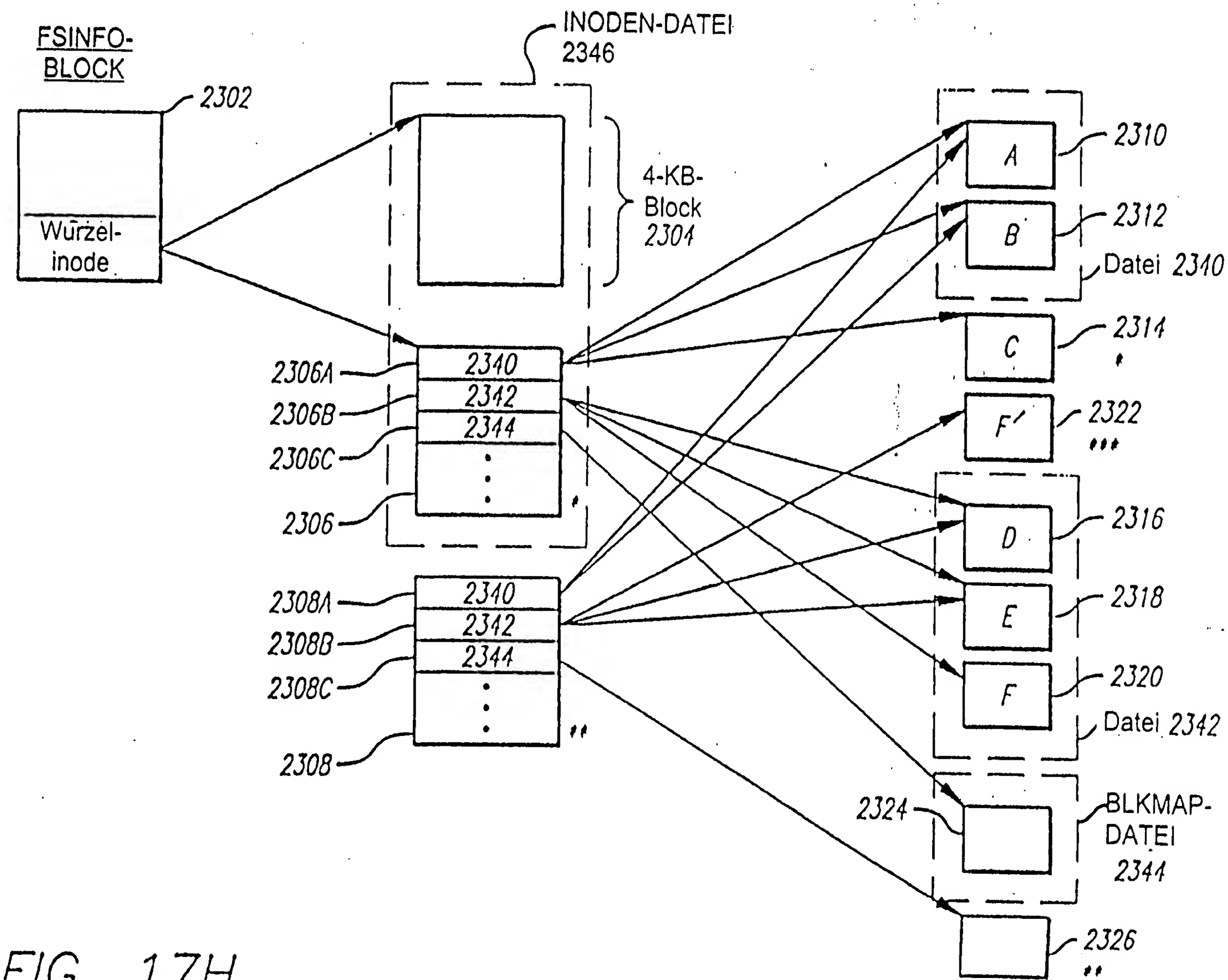

FSINFO-
BLOCK
2302
Wurzel-
inode
INODEN-DATEI
2346
4-KB-
Block
2304
2306A
2306B
2306C
2306
2340
2342
2344
2308A
2308B
2308C
2308
2340
2342
2344
A
B
C
F'
D
E
F
2310
2312
Datei 2340
2314
2322
2316
2318
2320
Datei 2342
2324
BLKMAP-
DATEI
2344
2326

FIG. 17H

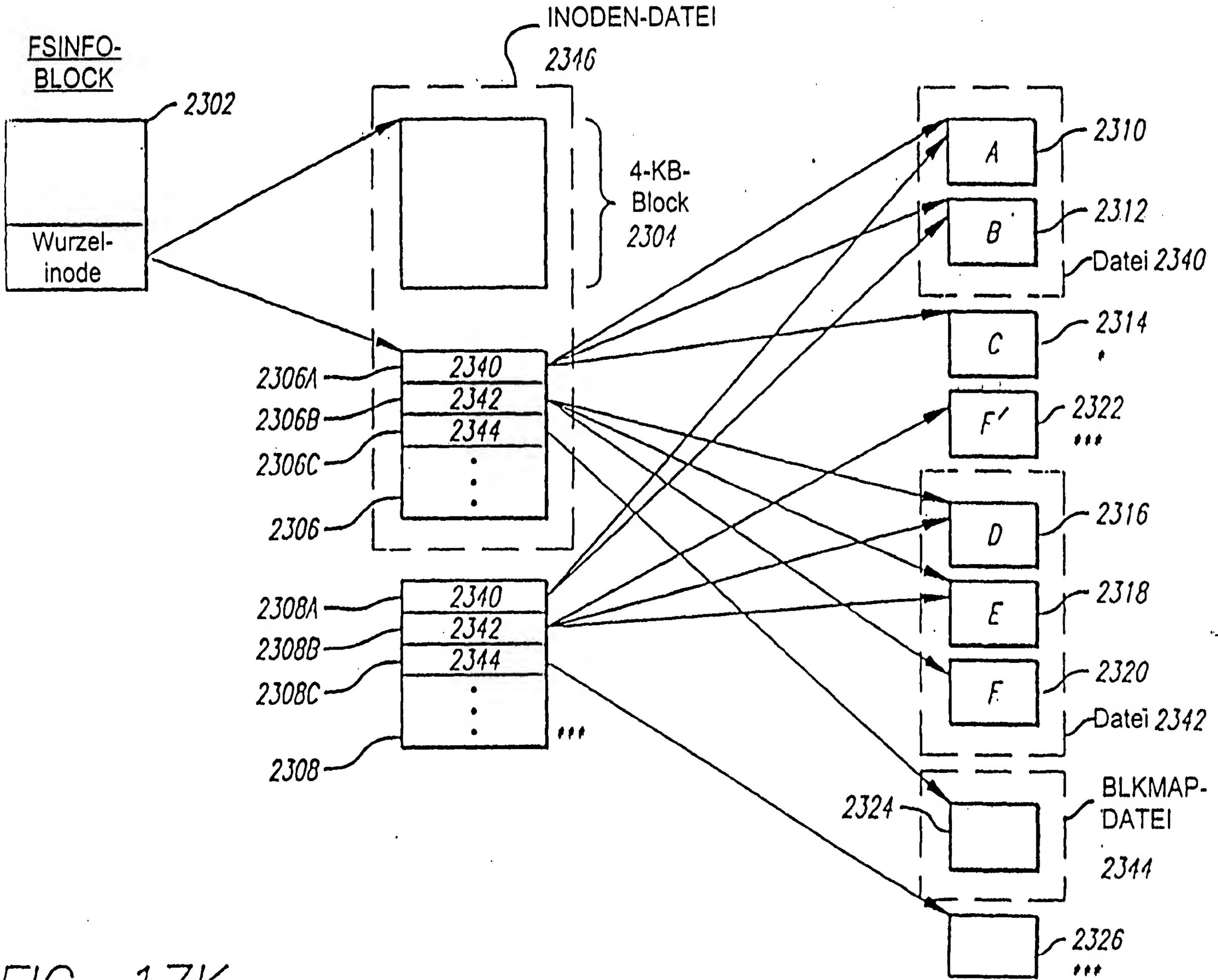

FSINFO-
BLOCK
2302
Wurzel-
inode
INODEN-DATEI
2346
4-KB-
Block
2301
2306A
2306B
2306C
2306
2340
2342
2344
2308A
2308B
2308C
2308
2340
2342
2344
A
B
C
F'
D
E
F
2310
2312
Datei 2340
2314
2322
2316
2318
2320
Datei 2342
2324
BLKMAP-
DATEI
2344
2326

FIG. 17K

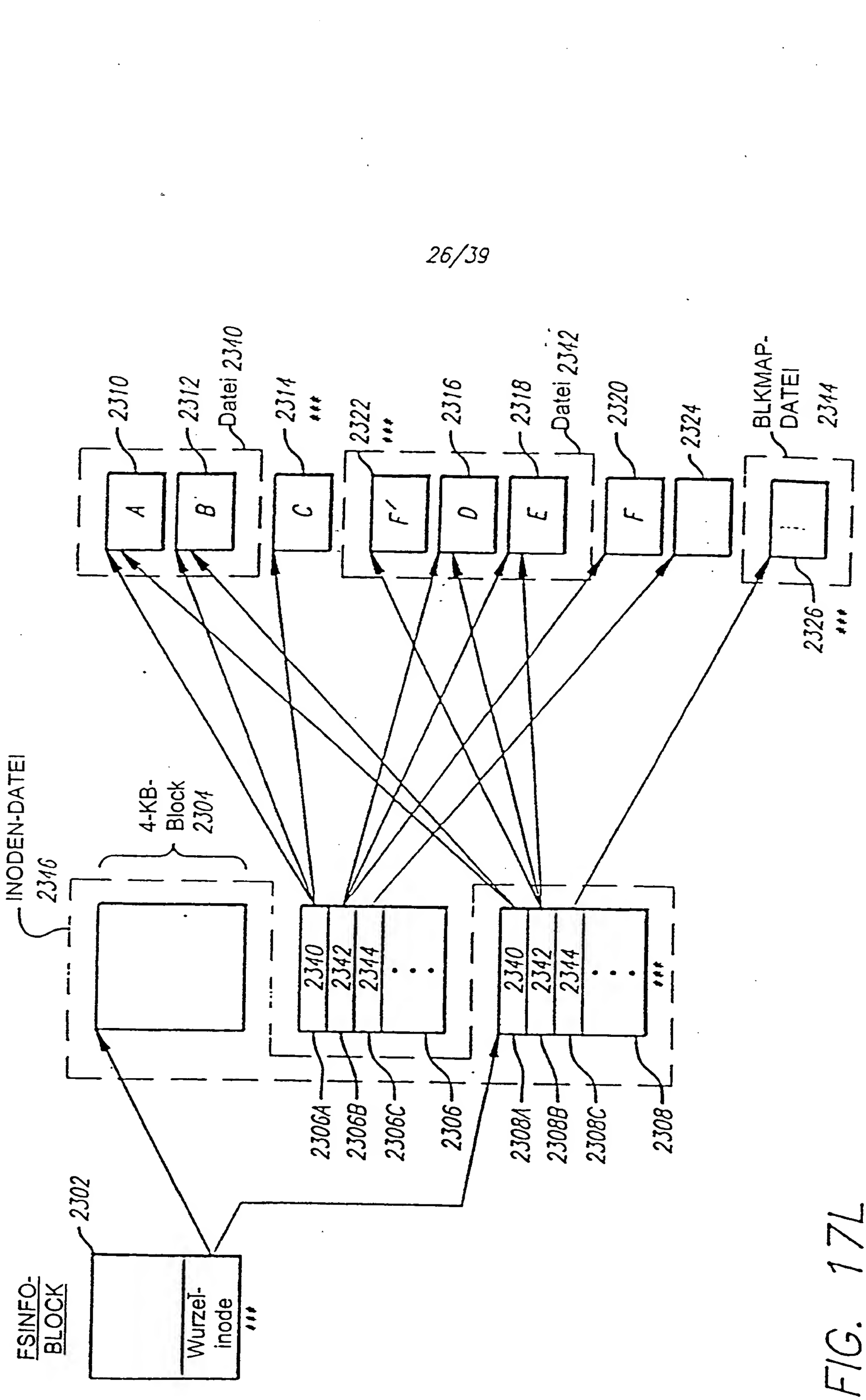
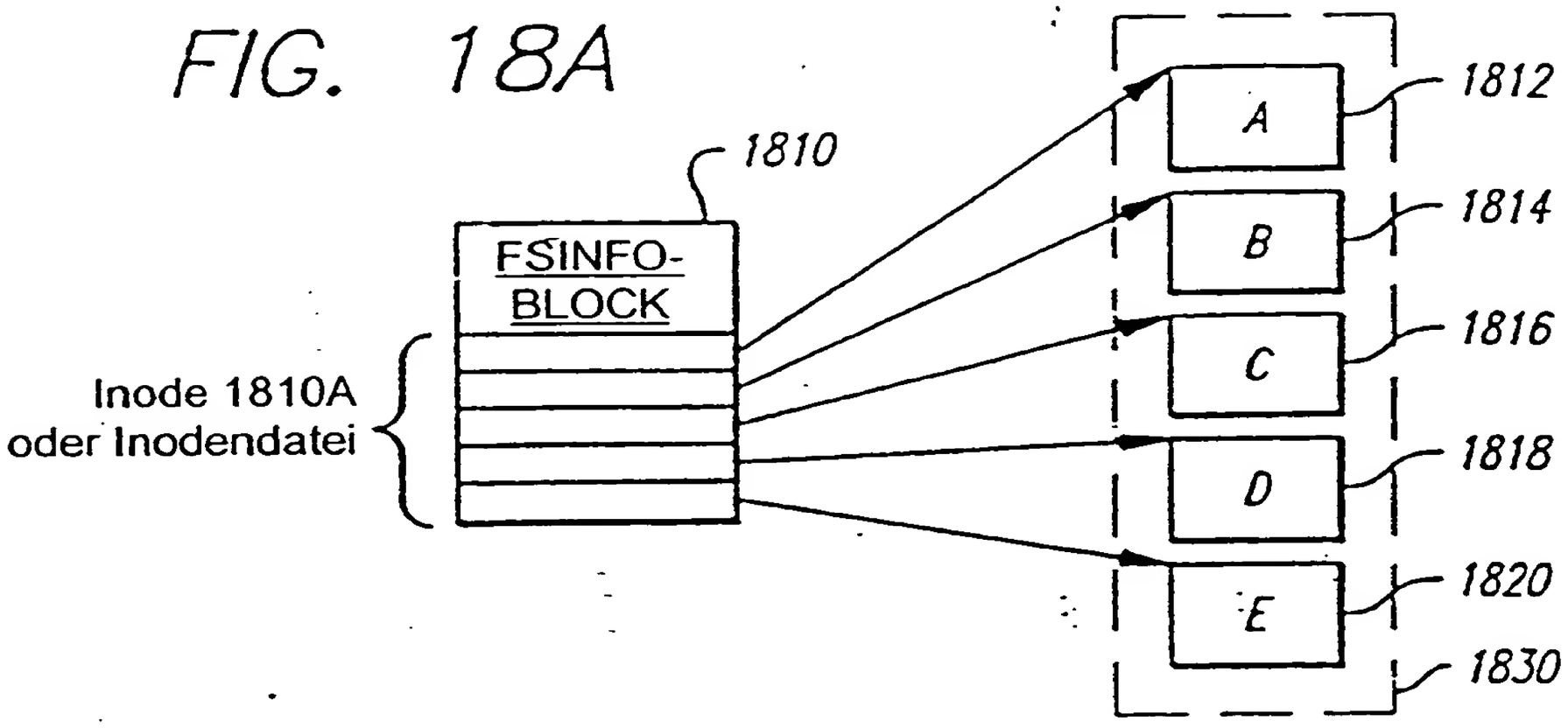

FSINFO-BLOCK
2302
Wurzel-inode
INODEN-DATEI
2346
4-KB-Block 2304
2306A
2306B
2306C
2306
2308A
2308B
2308C
2308
2310
2312
Datei 2340
2314
2316
2318
Datei 2342
2320
2322
2324
2326
BLKMAP-DATEI
2344
A
B
C
F'
D
E
F
2340
2342
2344

FIG. 17L

FIG. 18A



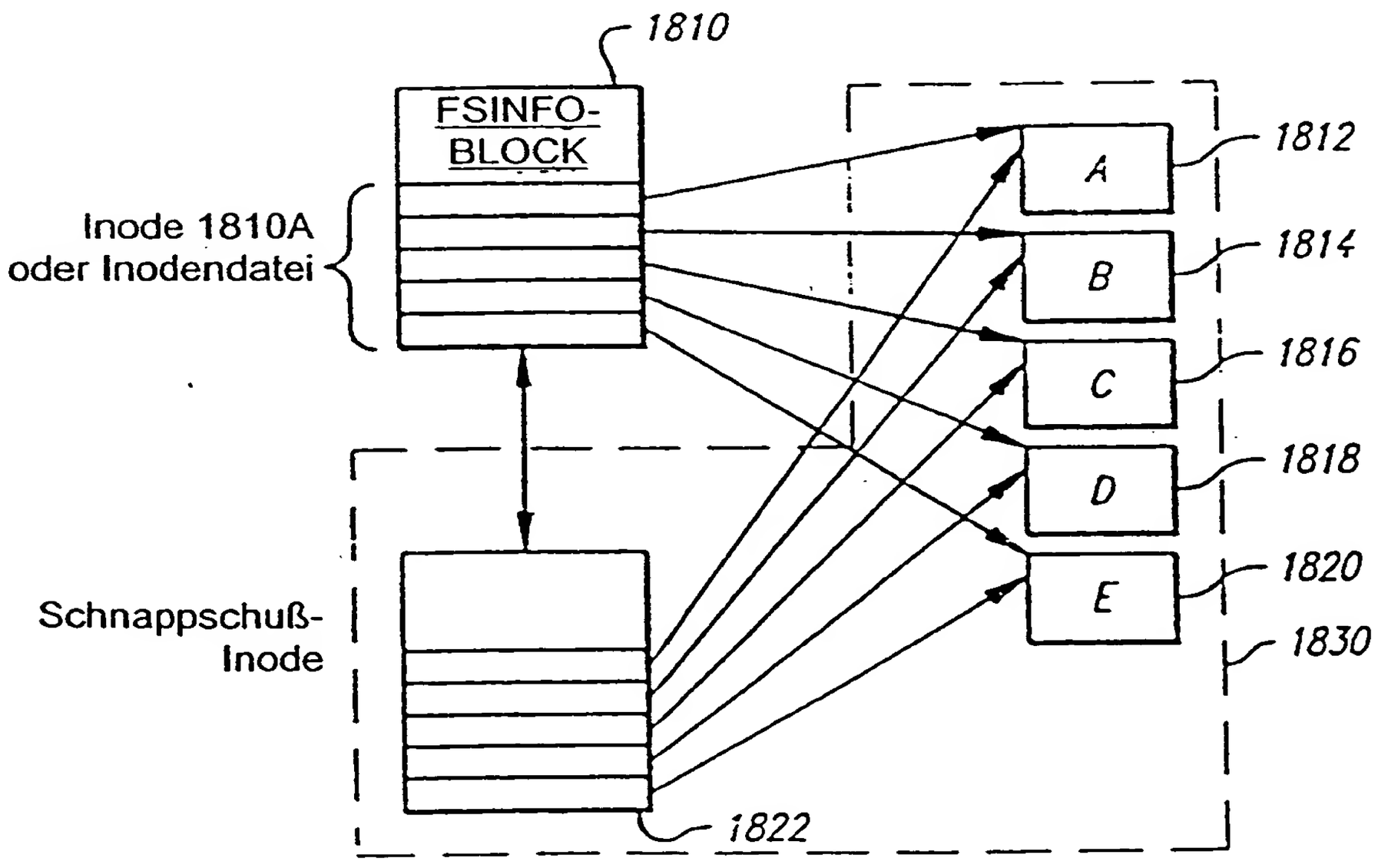
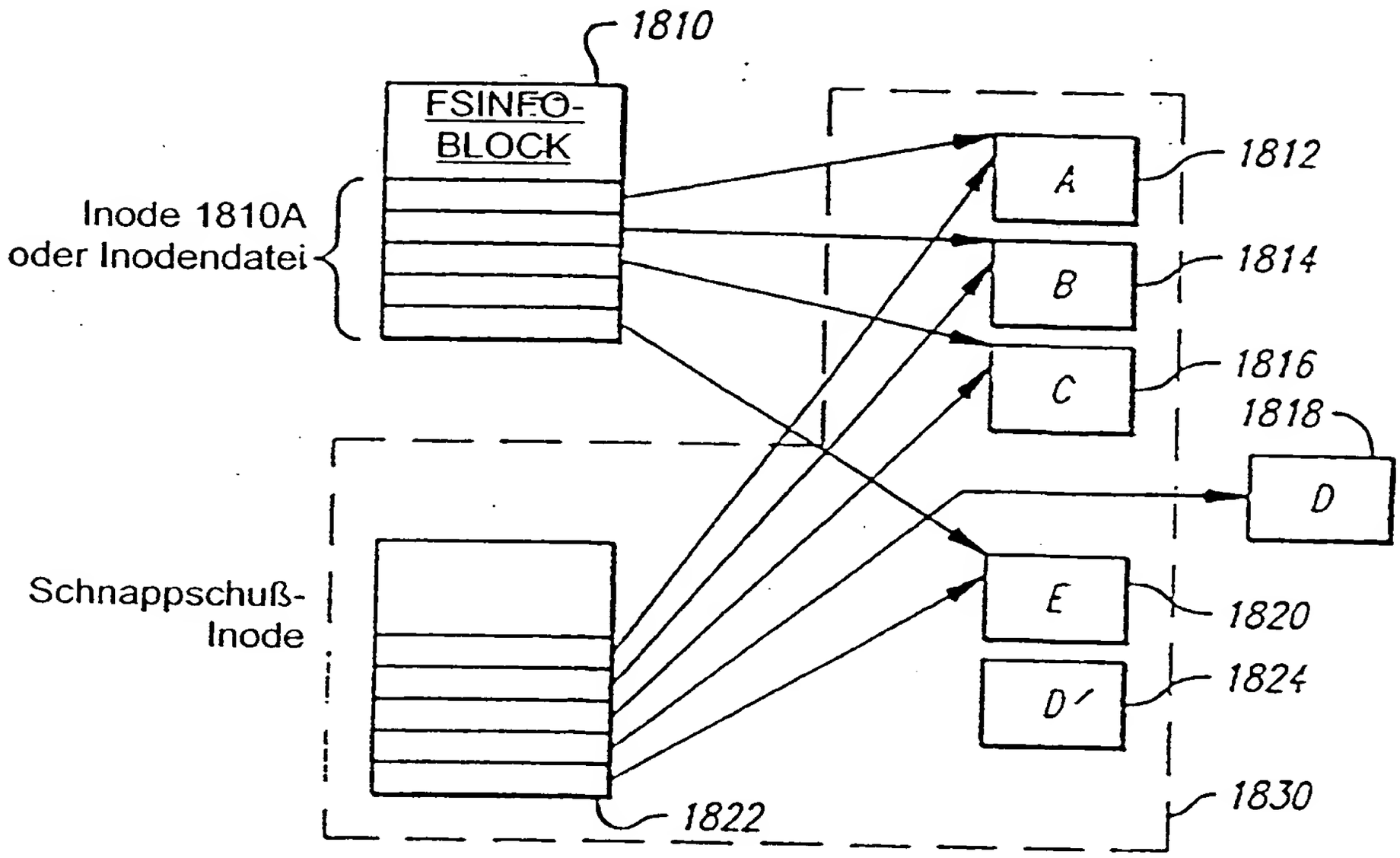


FIG. 18B

FIG. 18C


1810
FSINFO-
BLOCK
Inode 1810A
oder Inodendatei
A
B
C
D
E
D'
1812
1814
1816
1818
1820
1824
1830
Schnappschuß-
Inode
1822

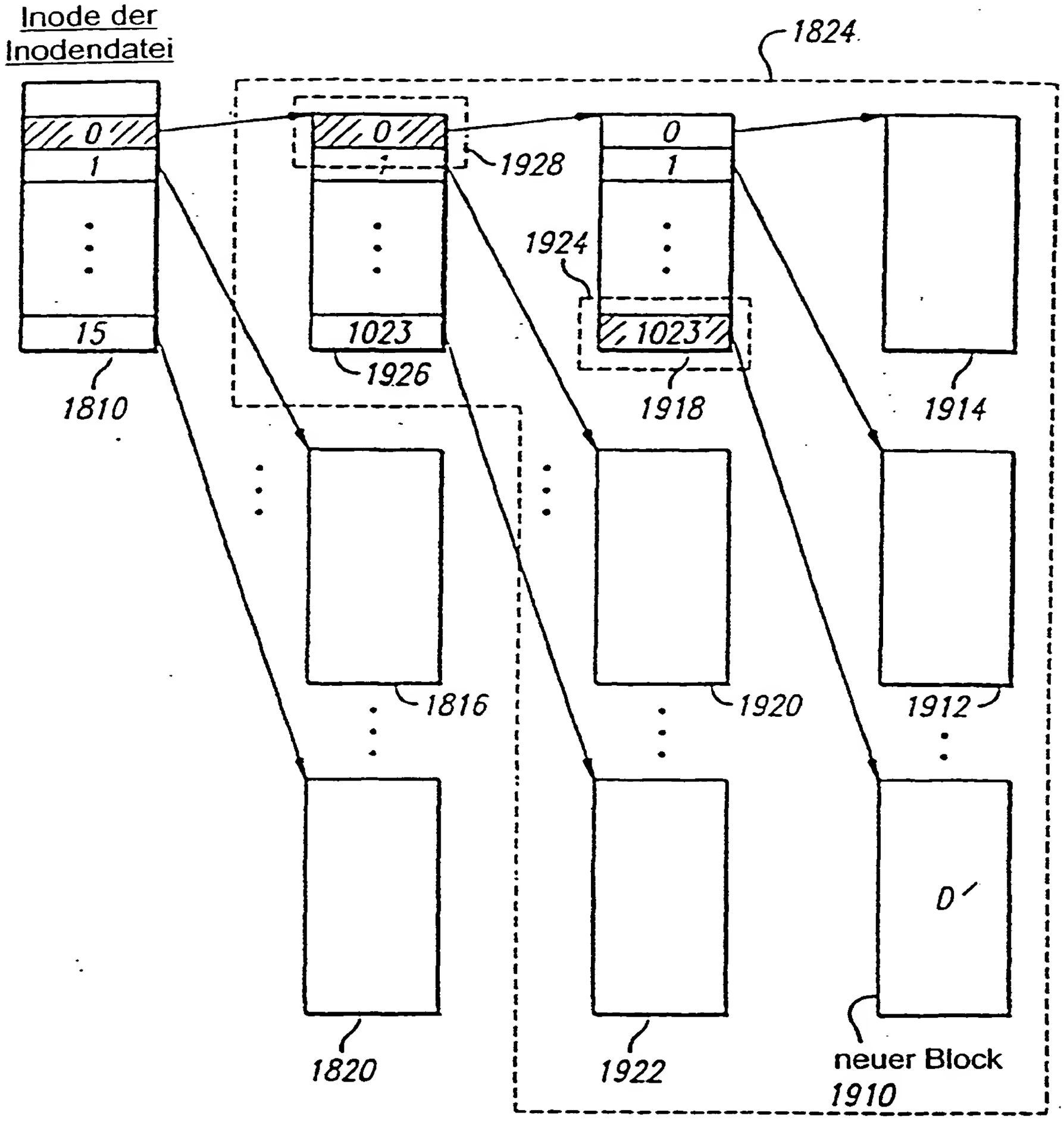

Inode der Inodendatei
1824
0
1
15
1810
0
1928
1023
1926
1816
1820
1924
0
1
1023
1918
1920
1922
1914
1912
D'
neuer Block
1910
doppelt indirekte Blöcke
einfach indirekte Blöcke
direkte Blöcke

FIG. 19

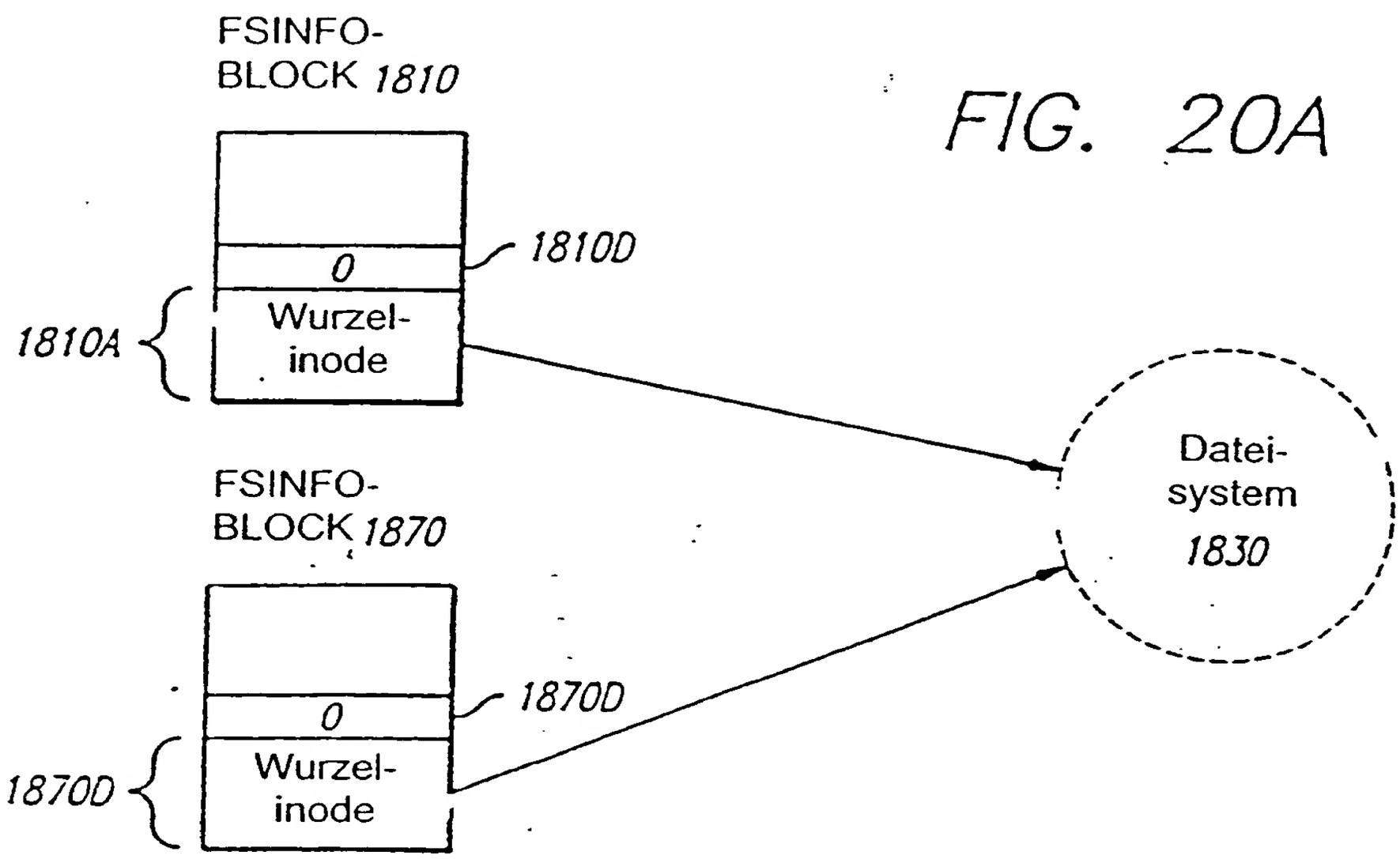


FIG. 20A

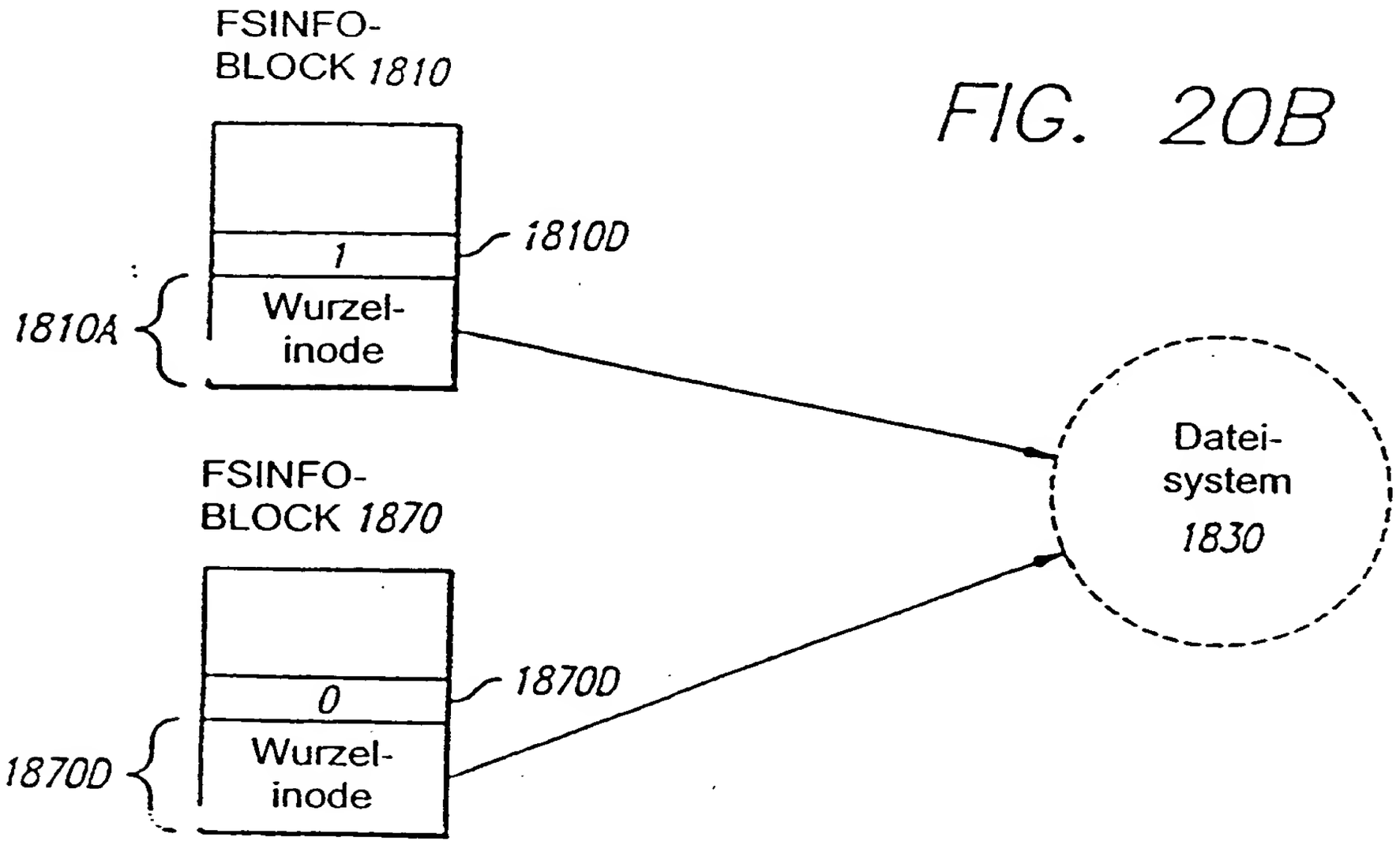


FIG. 20B

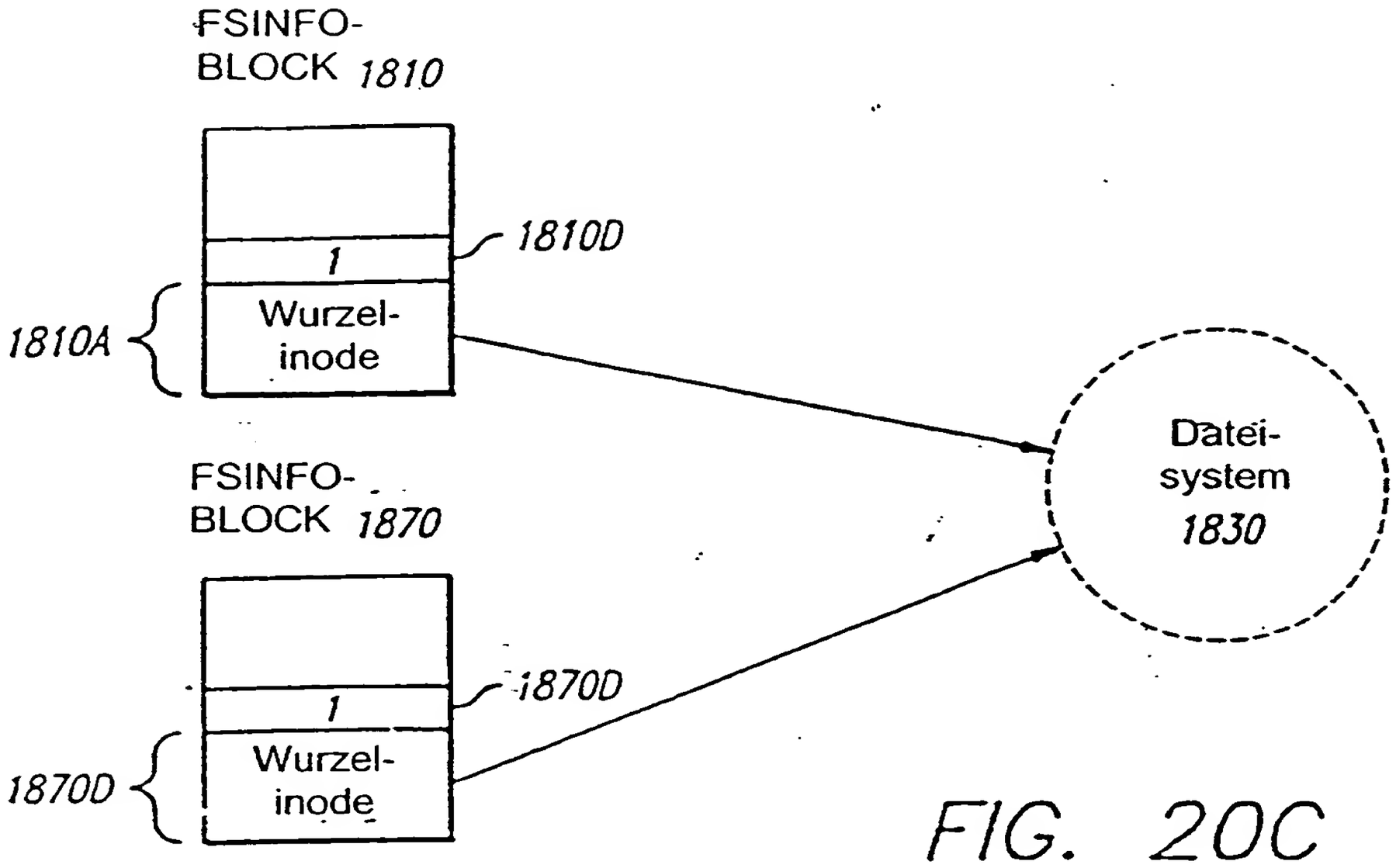


FIG. 20C

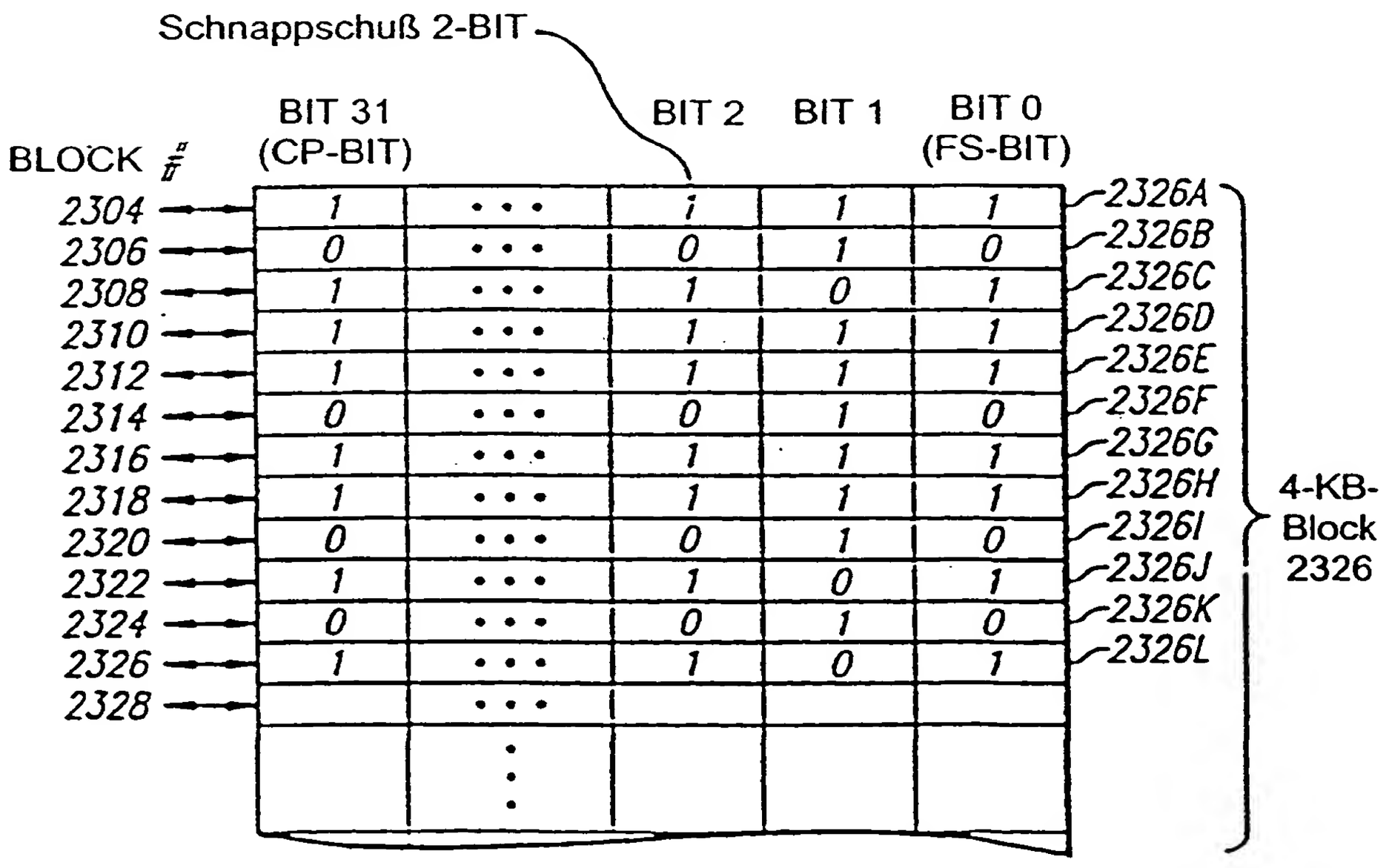


FIG. 21E

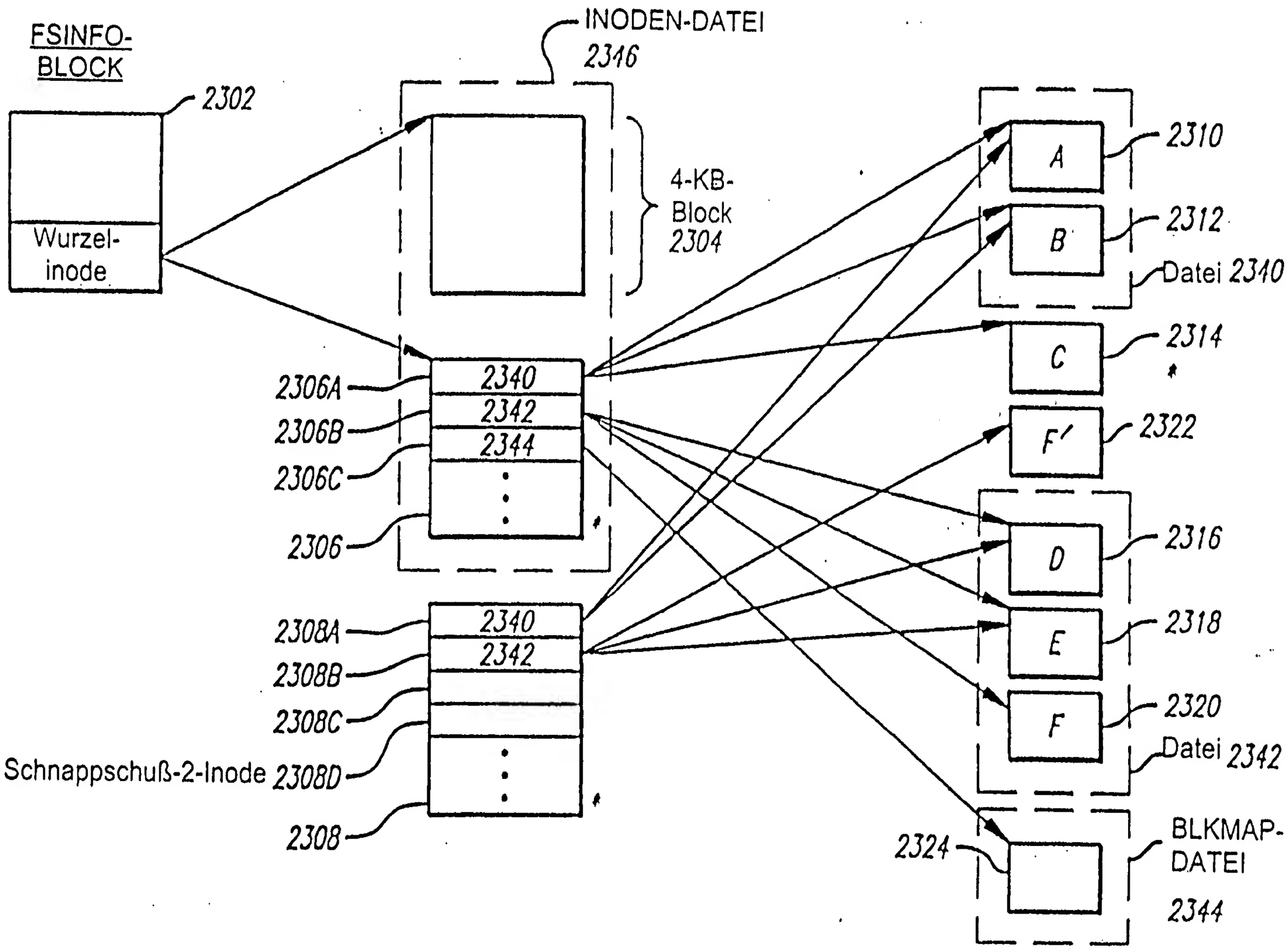

FSINFO-
BLOCK
2302
Wurzel-
inode
INODEN-DATEI
2346
4-KB-
Block
2304
2306A
2306B
2306C
2306
2340
2342
2344
2308A
2308B
2308C
2308D
2308
Schnappschuß-2-Inode
2340
2342
A
B
C
F'
D
E
F
2324
2310
2312
Datei 2340
2314
2322
2316
2318
2320
Datei 2342
BLKMAP-
DATEI
2344

FIG. 21A

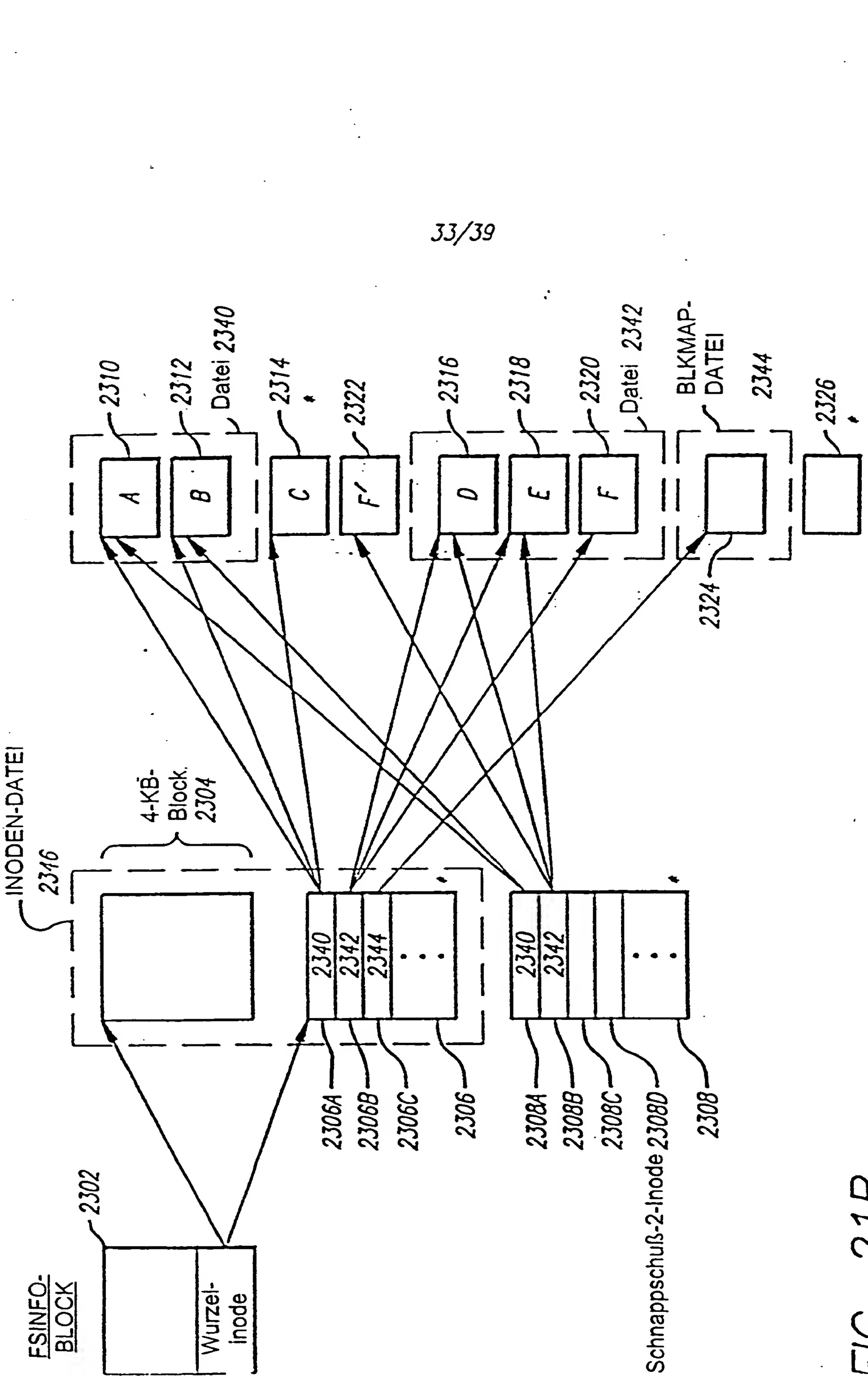

2310
2312
Datei 2340
2314
2322
2316
2318
2320
Datei 2342
BLKMAP-DATEI
2344
2326
A
B
C
F'
D
E
F
2324
INODEN-DATEI
2316
4-KB-Block 2304
2340
2342
2344
2340
2342
2306A
2306B
2306C
2306
2308A
2308B
2308C
Schnappschuß-2-Inode 2308D
2308
FSINFO-BLOCK
2302
Wurzel-inode

FIG. 21B

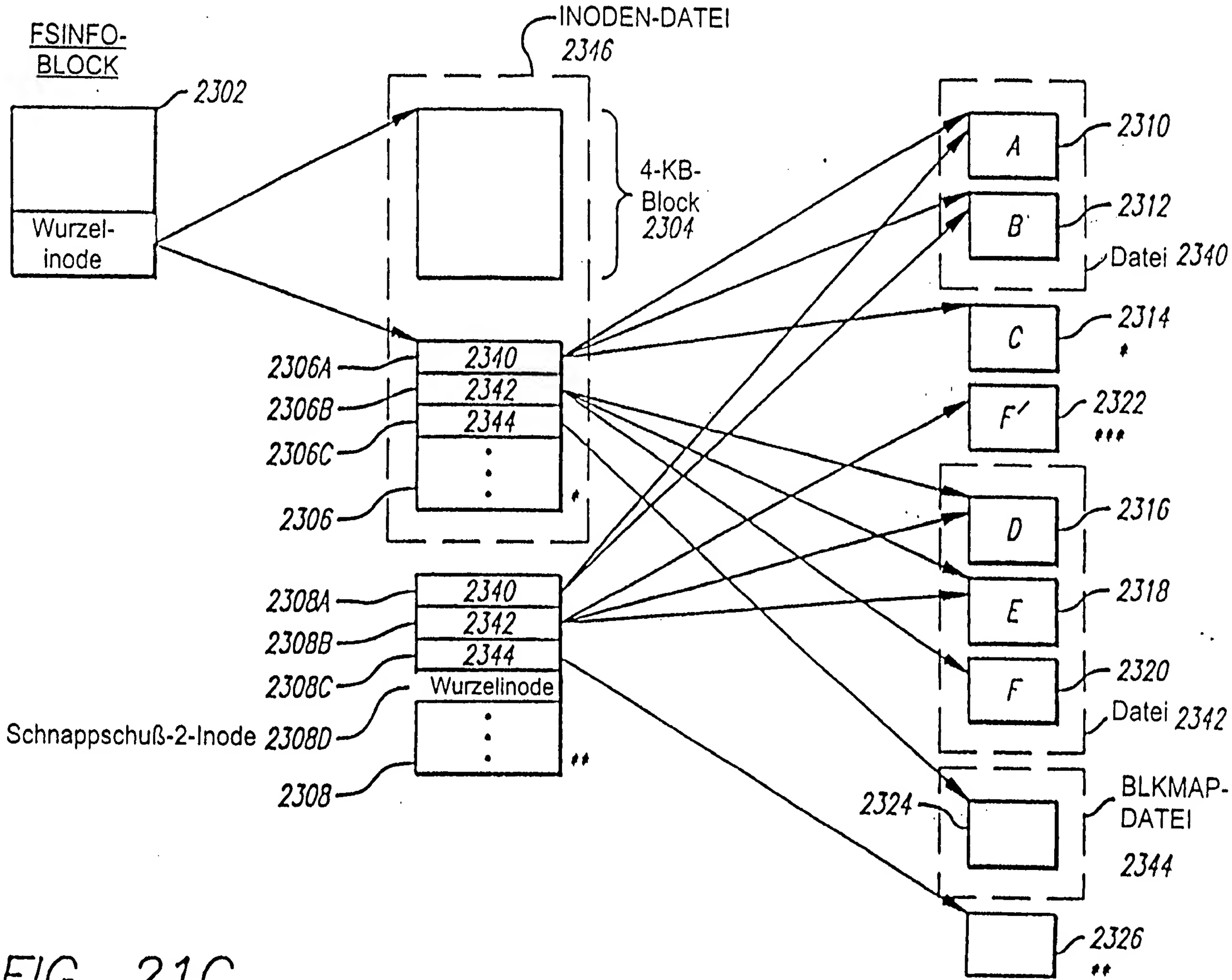

FSINFO-
BLOCK
2302
Wurzel-
inode
INODEN-DATEI
2316
4-KB-
Block
2304
2306A
2306B
2306C
2306
2340
2342
2344
2308A
2308B
2308C
2308D
2308
Schnappschuß-2-Inode
Wurzelinode
A
B
C
F'
D
E
F
2310
2312
Datei 2340
2314
2322
2316
2318
2320
Datei 2342
2324
BLKMAP-
DATEI
2344
2326

FIG. 21C

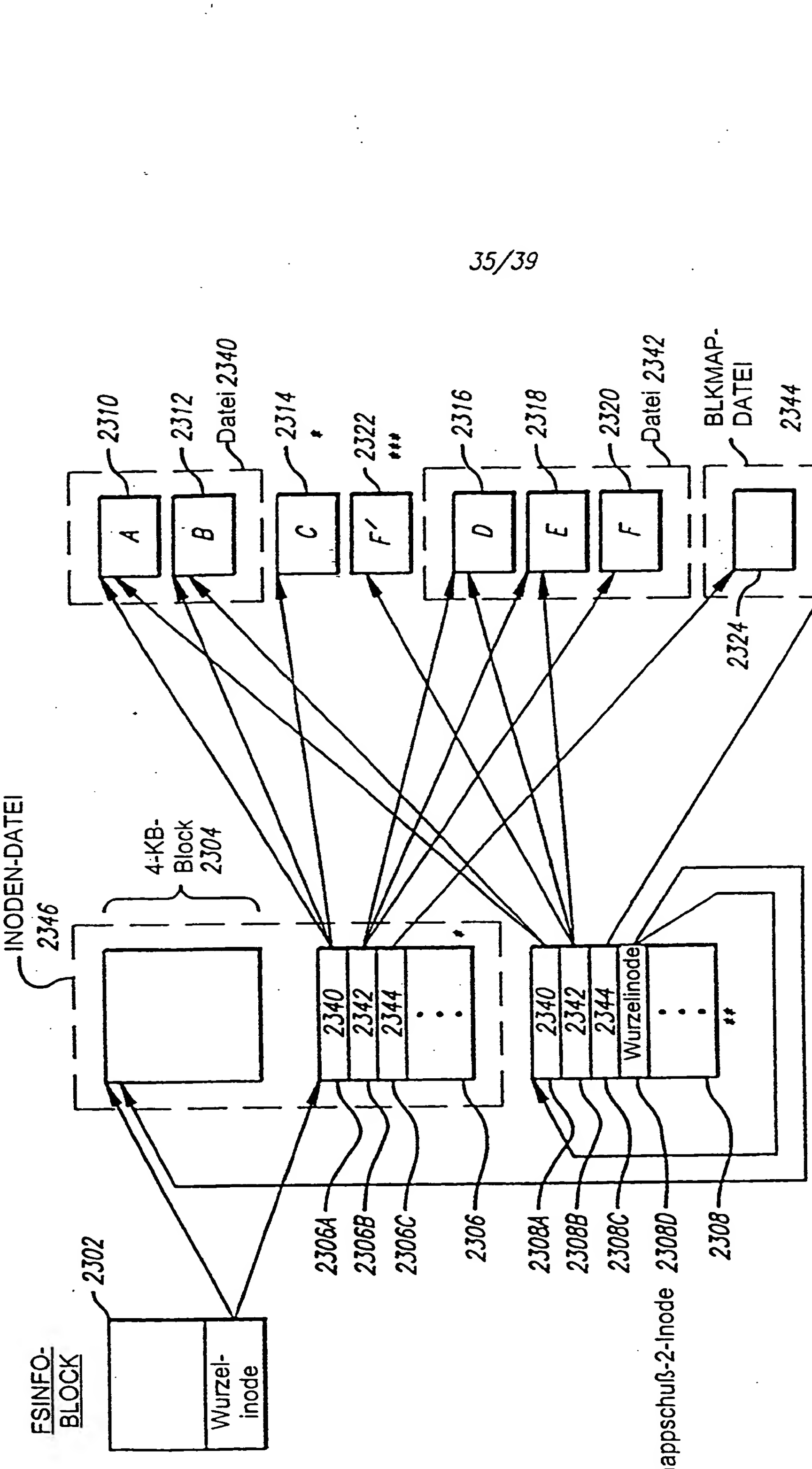

FSINFO-BLOCK
2302
Wurzel-inode
INODEN-DATEI
2346
4-KB-Block 2304
A
B
C
F'
D
E
F
2310
2312
Datei 2340
2314
2322
2316
2318
2320
Datei 2342
BLKMAP-DATEI
2344
2326
2324
2340
2342
2344
Wurzelinode
2306A
2306B
2306C
2306
2308A
2308B
2308C
Schnappschuß-2-Inode 2308D
2308

FIG. 21D

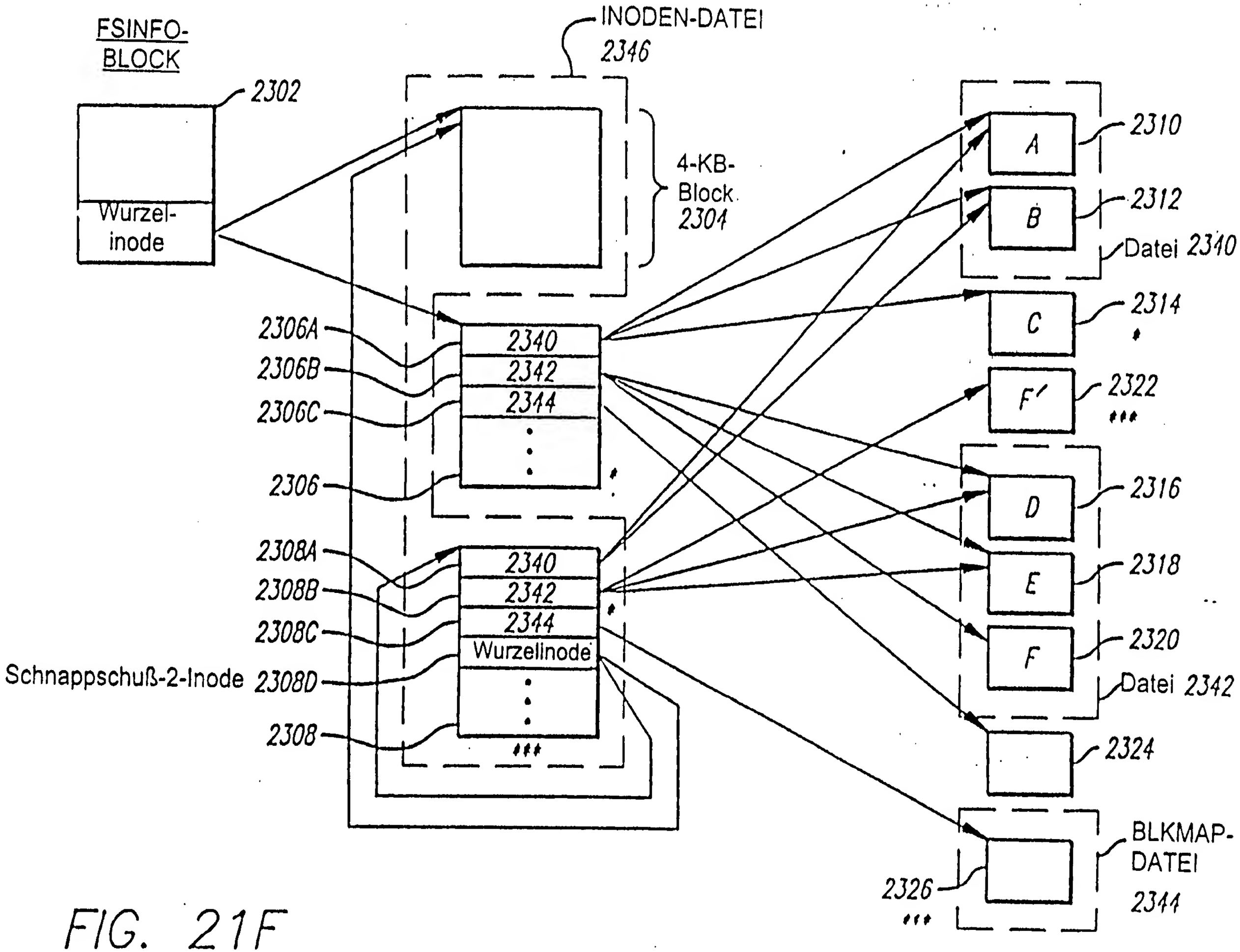

FSINFO-BLOCK
2302
Wurzel-inode
INODEN-DATEI
2346
4-KB-Block
2304
2306A
2306B
2306C
2306
2340
2342
2344
2308A
2308B
2308C
2308D
2308
Schnappschuß-2-Inode
Wurzelinode
A
B
C
F'
D
E
F
2310
2312
Datei 2340
2314
2322
2316
2318
2320
Datei 2342
2324
2326
BLKMAP-DATEI
2344

FIG. 21F

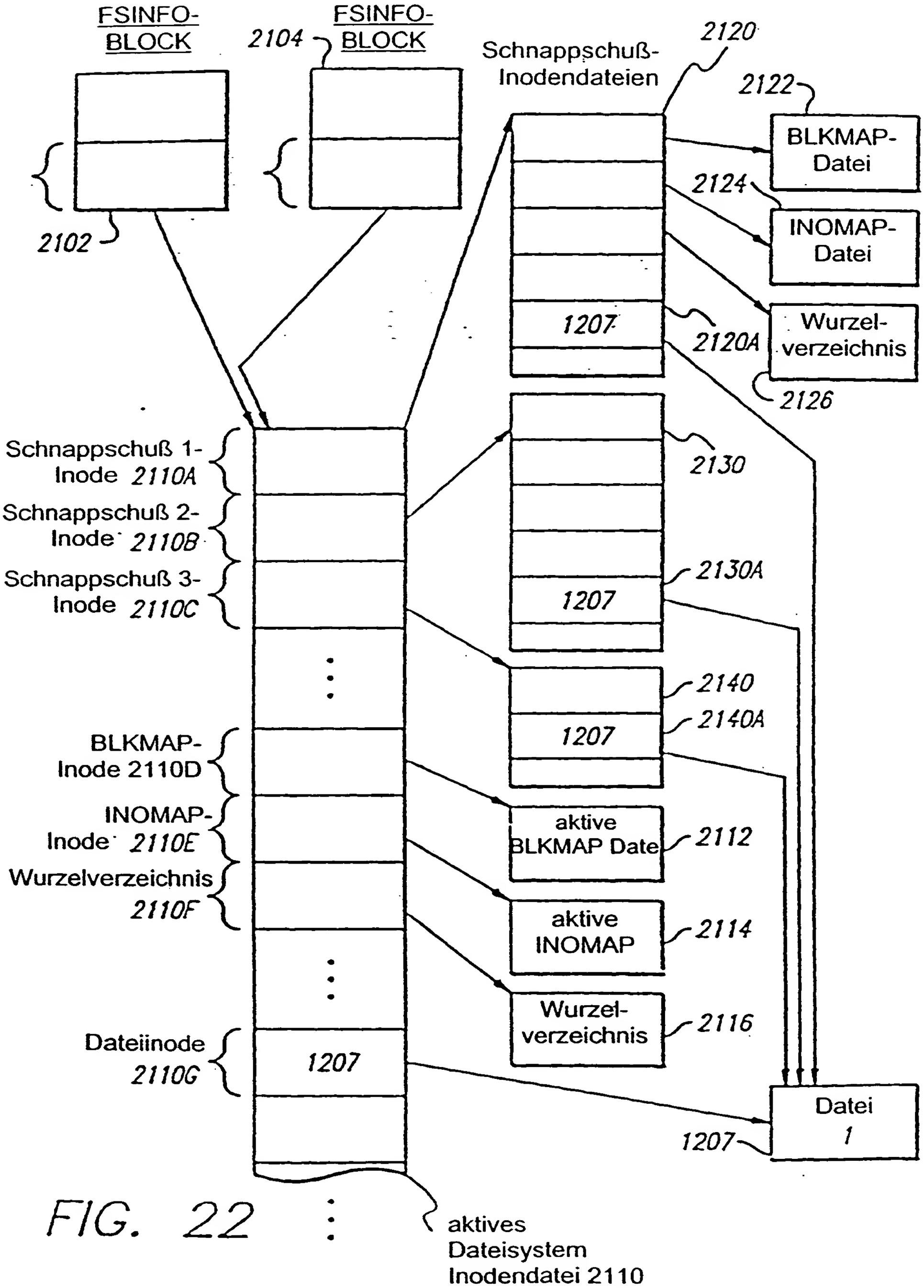

FSINFO-BLOCK
2104
FSINFO-BLOCK
2102
Schnappschuß-Inodendateien
2120
2122
BLKMAP-Datei
2124
INOMAP-Datei
Wurzel-verzeichnis
2126
2120A
1207
2130
2130A
1207
2140
2140A
1207
Schnappschuß 1-Inode 2110A
Schnappschuß 2-Inode 2110B
Schnappschuß 3-Inode 2110C
BLKMAP-Inode 2110D
INOMAP-Inode 2110E
Wurzelverzeichnis 2110F
Dateiinode 2110G
1207
aktive BLKMAP Date
2112
aktive INOMAP
2114
Wurzel-verzeichnis
2116
Datei 1
1207
aktives Dateisystem Inodendatei 2110

FIG. 22

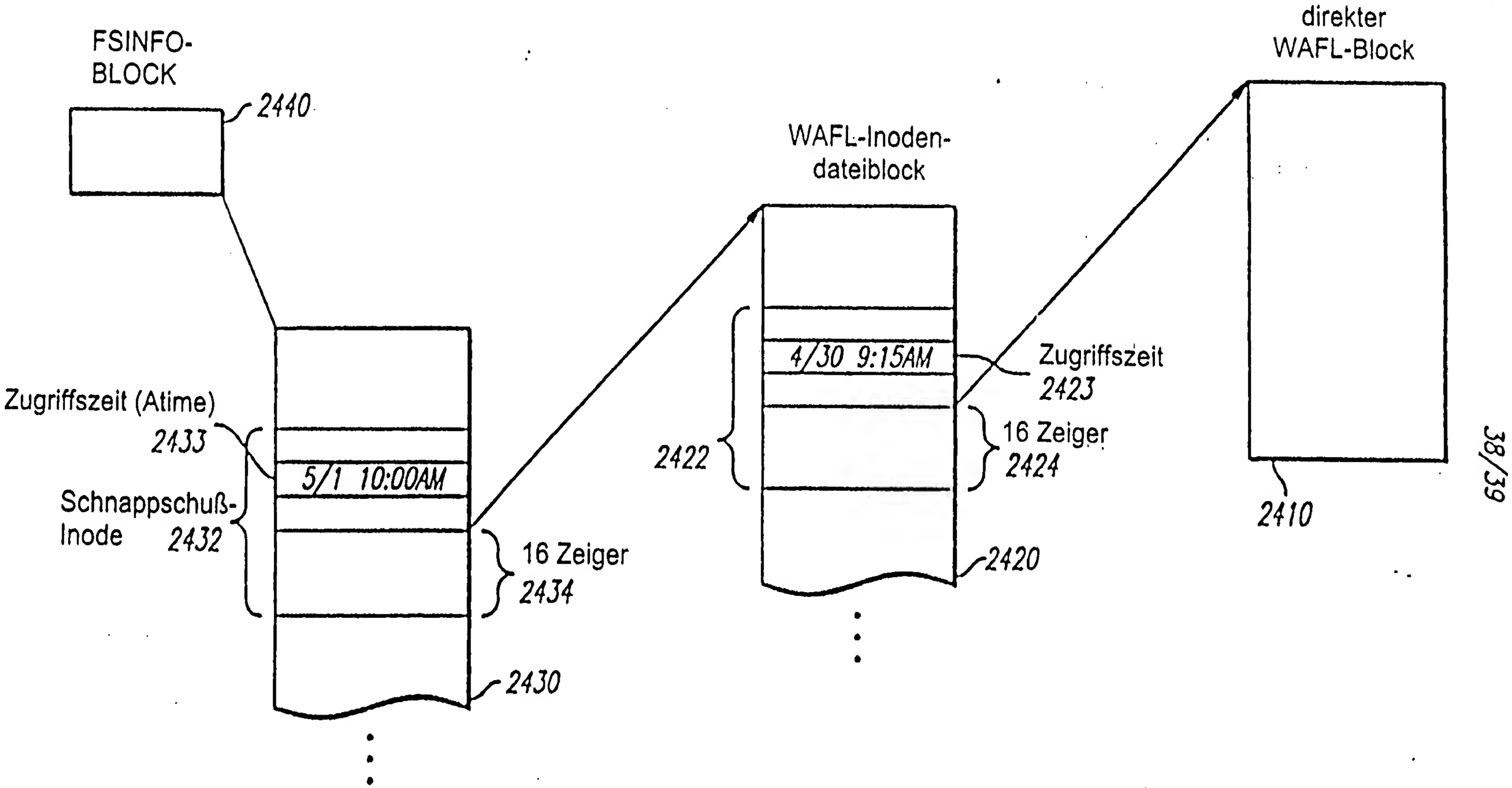

FSINFO-
BLOCK
2440
Zugriffszeit (Atime)
2433
5/1 10:00AM
Schnappschuß-
Inode
2432
16 Zeiger
2434
2430
WAFL-Inoden-
dateiblock
4/30 9:15AM
Zugriffszeit
2423
2422
16 Zeiger
2424
2420
direkter
WAFL-Block
2410

FIG. 23A

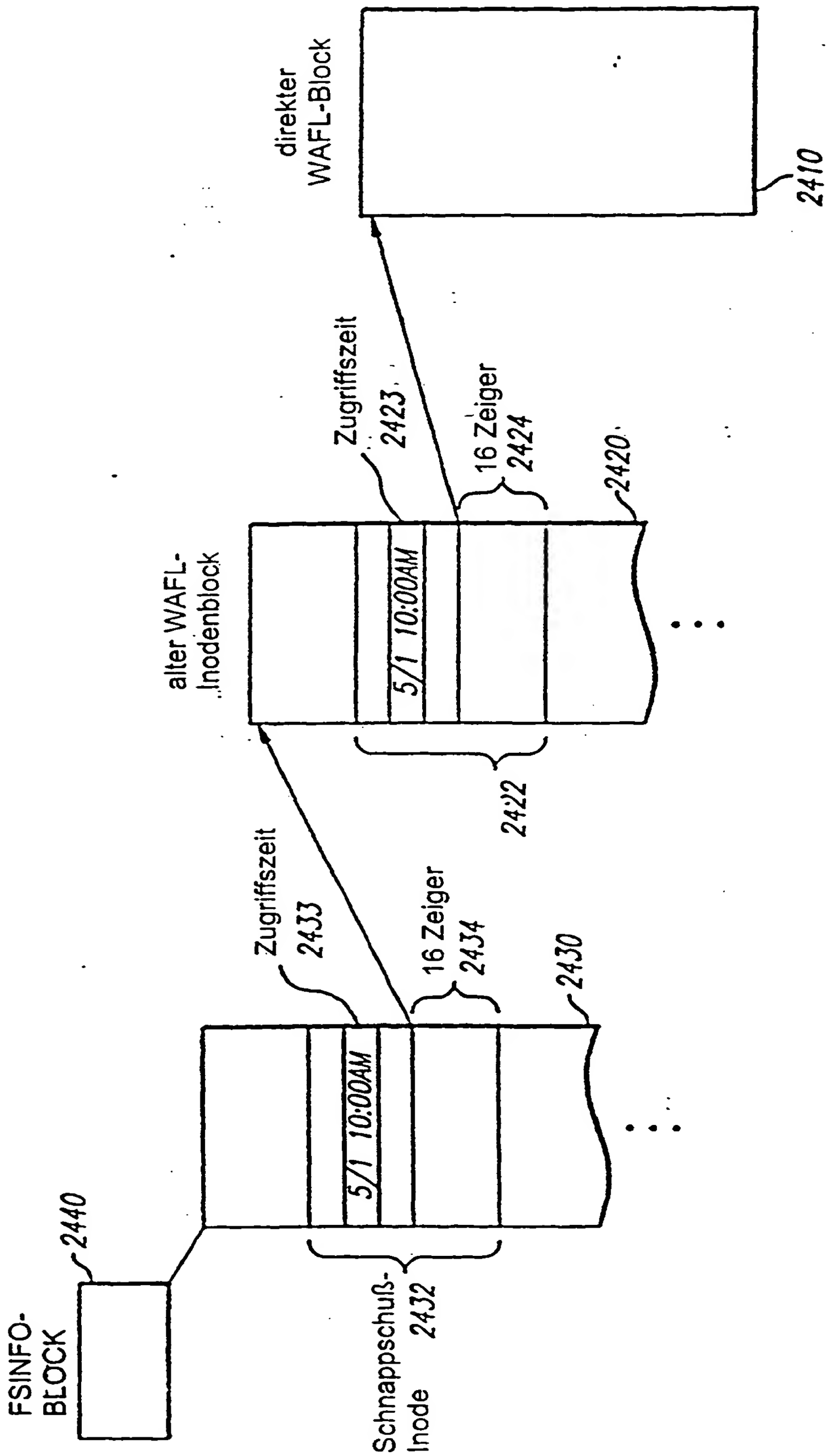

direkter WAFL-Block
2410
Zugriffszeit
2423
16 Zeiger
2424
2420
alter WAFL-Inodenblock
5/1 10:00AM
2422
Zugriffszeit
2433
16 Zeiger
2434
2430
FSINFO-BLOCK
2440
5/1 10:00AM
Schnappschuß-Inode 2432

FIG. 23B